DONALD F. ZIMMER, JR. (SBN 112279)
fzimmer@kslaw.com
CHERYL A. SABNIS (SBN 224323)
csabnis@kslaw.com
KING & SPALDING LLP
101 Second Street – Suite 2300
San Francisco, CA 94105
Telephone: (415) 318-1200
Facsimile: (415) 318-1300

IAN C. BALLON (SBN 141819)
ballon@gtlaw.com
HEATHER MEEKER (SBN 172148)
meekerh@gtlaw.com
GREENBERG TRAURIG, LLP
1900 University Avenue
East Palo Alto, CA 94303
Telephone: (650) 328-8500
Facsimile: (650) 328-8508

SCOTT T. WEINGAERTNER (*Pro Hac Vice*)
sweingaertner@kslaw.com
ROBERT F. PERRY
rperry@kslaw.com
BRUCE W. BABER (*Pro Hac Vice*)
bbaber@kslaw.com
KING & SPALDING LLP
1185 Avenue of the Americas
New York, NY 10036-4003
Telephone: (212) 556-2100
Facsimile: (212) 556-2222

ROBERT A. VAN NEST - #84065
rvannest@kvn.com
CHRISTA M. ANDERSON - #184325
canderson@kvn.com
KEKER & VAN NEST LLP
710 Sansome Street
San Francisco, CA 94111-1704
Telephone: (415) 391-5400
Facsimile: (415) 397-7188

Attorneys for Defendant
GOOGLE INC.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN FRANCISCO DIVISION

ORACLE AMERICA, INC.

Plaintiff,

v.

GOOGLE INC.

Defendant.

Case No. 3:10-cv-03561-WHA

Honorable Judge William Alsup

DECLARATION OF SCOTT T. WEINGAERTNER IN SUPPORT OF GOOGLE, INC.'S DAUBERT MOTION

HIGHLY CONFIDENTIAL -- ATTORNEYS' EYES ONLY

I, Scott T. Weingaertner, declare as follows:

I am a partner in the law firm of King & Spalding LLP, counsel to Google Inc. in the present case. I submit this declaration in support of the Google Inc.'s Daubert Motion. I make this declaration based on my own personal knowledge. If called as a witness, I could and would testify competently to the matters set forth herein.

1. Attached to this declaration as Exhibit A is a true and correct copy of the Expert Report of Iain M. Cockburn (including exhibits and appendices), served by Oracle America, Inc. ("Oracle") on May 21, 2011. **[FILED UNDER SEAL]**

2. Attached to this declaration as Exhibit B is a true and correct copy of Oracle's Technology Tutorial Supplement, dated April 6, 2011.

3. Attached to this declaration as Exhibit C is a true and correct copy of the cover document of Oracle's Second Supplemental Patent Local Rule 3-1 Disclosure of Asserted Claims and Infringement Contentions ("Oracle's Infringement Contentions"), served by Oracle on April 1, 2011.

4. Attached to this declaration as Exhibit D is a true and correct copy of Exhibit D to Oracle's Infringement Contentions, served by Oracle on April 1, 2011.

5. Attached to this declaration as Exhibit E is a true and correct copy of Exhibit G to Oracle's Infringement Contentions, served by Oracle on April 1, 2011.

6. Attached to this declaration as Exhibit F is a true and correct copy of Defendant Google Inc.'s Fourth Supplemental Responses to Plaintiff's Interrogatories, Set One, No. 3, served by Google Inc. ("Google") on April 27, 2011.

7. Attached to this declaration as Exhibit G is a true and correct copy of an Android Native Development Kit webpage, downloaded from http://developer.android.com/sdk/ndk/index.html on June 14, 2011.

8. Attached to this declaration as Exhibit H is a true and correct copy of OAGOOGLE0000140295 - OAGOOGLE0000140499, entitled "Form CO relating to the notification of a concentration under Council Regulation (EC) No. 139/2004." **[FILED UNDER SEAL]**

9. Attached to this declaration as Exhibit I is a true and correct copy of OAGOOGLE0002796883, a spreadsheet originally produced in its native format by Oracle. **[FILED UNDER SEAL]**

10. Attached to this declaration as Exhibit J is a true and correct copy of OAGOOGLE0100030742 - OAGOOGLE0100031130, entitled "Oracle Corporation Estimation of the Fair Value of Certain Assets and Liabilities of Sun Microsystems, inc. as of January 26, 2010." **[FILED UNDER SEAL]**

11. Attached to this declaration as Exhibit K is a true and correct copy of OAGOOGLE0000062503 - OAGOOGLE0000062726, Oracle Corporation's Form 10-K, filed with the U.S. Securities and Exchange Commission on July 1, 2010.

12. Attached to this declaration as Exhibit L is a true and correct copy of excerpts from OAGOOGLE0000062097, a spreadsheet originally produced in its native format by Oracle. **[FILED UNDER SEAL]**.

13. Attached to this declaration as Exhibit M is a true and correct copy of OAGOOGLE0100071840 - OAGOOGLE0100071986, entitled "SW OEM Pricebook." **[FILED UNDER SEAL]**

14. Attached to this declaration as Exhibit N is a true and correct copy of a January 23, 2001 news press release entitled "Microsoft Reaches Agreement to Settle Contract Dispute With Sun Microsystems," downloaded from http://www.microsoft.com/presspass/press/2001/jan01/01-23sunpr.mspx on June 14, 2011.

15. Attached to this declaration as Exhibit O is a true and correct copy of a Settlement Agreement and Mutual Limited Release downloaded from http://www.microsoft.com/presspass/legal/01-23settlement.mspx on June 14, 2011.

16. Attached to this declaration as Exhibit P is a true and correct copy of OAGOOGLE0100003277 - OAGOOGLE0100003291, a "Stand-Alone TCK License Agreement" entered into by Sun Microsystems, Inc. and Oracle Corporation on March 25, 2004. **[FILED UNDER SEAL]**

17. Attached to this declaration as Exhibit Q is a true and correct copy of OAGOOGLE0100005211 - OAGOOGLE0100005221, a “Stand-Alone TCK License Agreement” entered into by Sun Microsystems, Inc. and SAP AG on May 16, 2005. **[FILED UNDER SEAL]**

18. Attached to this declaration as Exhibit R is a true and correct copy of OAGOOGLE0100165699 - OAGOOGLE0100165746, a July 6, 2010 Oracle presentation entitled “Q1 FY11 Java Sales Review.” **[FILED UNDER SEAL]**

19. Attached to this declaration as Exhibit S is a true and correct copy of an Android timeline, downloaded from http://www.android.com/timeline.html on June 14, 2011.

20. Attached to this declaration as Exhibit T is a true and correct copy of http://www.javaworld.com/javaworld/jw-02-2011/110204-android-market.html, downloaded on June 14, 2011.

21. Attached to this declaration as Exhibit U is a true and correct copy of http://www.betanews.com/article/Google-unveils-10-huge-improvements-in-FroYo-Android-22/1274374860, downloaded on June 14, 2011.

22. Attached to this declaration as Exhibit V is a true and correct copy of http://www.tabletsquad.com/top-5-improvements-in-android-3-0/, downloaded f on June 14, 2011.

23. Attached to this declaration as Exhibit W is a true and correct copy of OAGOOGLE0000140115 - OAGOOGLE0000140130, a March 12, 2009 letter from Oracle Corporation Chief Executive Officer Lawrence J. Ellison to the Sun Microsystems, Inc. Board of Directors. **[FILED UNDER SEAL]**

24. Attached to this declaration as Exhibit X is a true and correct copy of http://discussion.forum.nokia.com/forum/showthread.php?11133-j2me-compatibility-between-different-manufacuturers, downloaded f on June 14, 2011.

25. Attached to this declaration as Exhibit Y is a true and correct copy of http://www.russellbeattie.com/blog/1005717, downloaded on June 14, 2011.

26. Attached to this declaration as Exhibit Z is a true and correct copy of http://www.odi.ch/weblog/posting.php?posting=135, downloaded on June 14, 2011.

27. Attached to this declaration as Exhibit AA is a true and correct copy of http://www.oracle.com/technetwork/articles/javame/stateoftheunion-138337.html, downloaded on June 14, 2011.

28. Attached to this declaration as Exhibit BB is a true and correct copy of http://news.cnet.com/8301-13580_3-9800679-39.html?part=rss&subj=news&tag=2547-1_3-0-20, downloaded on June 14, 2011.

29. Attached to this declaration as Exhibit CC is a true and correct copy of http://portal.acm.org/citation.cfm?id=1839348, downloaded on June 14, 2011.

I declare under penalty of perjury that the foregoing facts are true and correct.

Executed on June 14, 2011 in New York, New York.

/s/ Scott T. Weingaertner /s/
Scott T. Weingaertner

DONALD F. ZIMMER, JR. (SBN 112279)
fzimmer@kslaw.com
CHERYL A. SABNIS (SBN 224323)
csabnis@kslaw.com
KING & SPALDING LLP
101 Second Street – Suite 2300
San Francisco, CA 94105
Telephone: (415) 318-1200
Facsimile: (415) 318-1300

IAN C. BALLON (SBN 141819)
ballon@gtlaw.com
HEATHER MEEKER (SBN 172148)
meekerh@gtlaw.com
GREENBERG TRAURIG, LLP
1900 University Avenue
East Palo Alto, CA 94303
Telephone: (650) 328-8500
Facsimile: (650) 328-8508

SCOTT T. WEINGAERTNER (*Pro Hac Vice*)
sweingaertner@kslaw.com
ROBERT F. PERRY
rperry@kslaw.com
BRUCE W. BABER (*Pro Hac Vice*)
bbaber@kslaw.com
KING & SPALDING LLP
1185 Avenue of the Americas
New York, NY 10036-4003
Telephone: (212) 556-2100
Facsimile: (212) 556-2222

ROBERT A. VAN NEST - #84065
rvannest@kvn.com
CHRISTA M. ANDERSON - #184325
canderson@kvn.com
KEKER & VAN NEST LLP
710 Sansome Street
San Francisco, CA 94111-1704
Telephone: (415) 391-5400
Facsimile: (415) 397-7188

Attorneys for Defendant
GOOGLE INC.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN FRANCISCO DIVISION

ORACLE AMERICA, INC.

Plaintiff,

v.

GOOGLE INC.

Defendant.

Case No. 3:10-cv-03561-WHA

Honorable Judge William Alsup

DECLARATION OF SCOTT T. WEINGAERTNER IN SUPPORT OF GOOGLE, INC.'S DAUBERT MOTION

HIGHLY CONFIDENTIAL -- ATTORNEYS' EYES ONLY

I, Scott T. Weingaertner, declare as follows:

I am a partner in the law firm of King & Spalding LLP, counsel to Google Inc. in the present case. I submit this declaration in support of the Google Inc.'s Daubert Motion. I make this declaration based on my own personal knowledge. If called as a witness, I could and would testify competently to the matters set forth herein.

1. Attached to this declaration as Exhibit A is a true and correct copy of the Expert Report of Iain M. Cockburn (including exhibits and appendices), served by Oracle America, Inc. ("Oracle") on May 21, 2011. **[FILED UNDER SEAL]**

2. Attached to this declaration as Exhibit B is a true and correct copy of Oracle's Technology Tutorial Supplement, dated April 6, 2011.

3. Attached to this declaration as Exhibit C is a true and correct copy of the cover document of Oracle's Second Supplemental Patent Local Rule 3-1 Disclosure of Asserted Claims and Infringement Contentions ("Oracle's Infringement Contentions"), served by Oracle on April 1, 2011.

4. Attached to this declaration as Exhibit D is a true and correct copy of Exhibit D to Oracle's Infringement Contentions, served by Oracle on April 1, 2011.

5. Attached to this declaration as Exhibit E is a true and correct copy of Exhibit G to Oracle's Infringement Contentions, served by Oracle on April 1, 2011.

6. Attached to this declaration as Exhibit F is a true and correct copy of Defendant Google Inc.'s Fourth Supplemental Responses to Plaintiff's Interrogatories, Set One, No. 3, served by Google Inc. ("Google") on April 27, 2011.

7. Attached to this declaration as Exhibit G is a true and correct copy of an Android Native Development Kit webpage, downloaded from http://developer.android.com/sdk/ndk/index.html on June 14, 2011.

8. Attached to this declaration as Exhibit H is a true and correct copy of OAGOOGLE0000140295 - OAGOOGLE0000140499, entitled "Form CO relating to the notification of a concentration under Council Regulation (EC) No. 139/2004." **[FILED UNDER SEAL]**

9. Attached to this declaration as Exhibit I is a true and correct copy of OAGOOGLE0002796883, a spreadsheet originally produced in its native format by Oracle. **[FILED UNDER SEAL]**

10. Attached to this declaration as Exhibit J is a true and correct copy of OAGOOGLE0100030742 - OAGOOGLE0100031130, entitled "Oracle Corporation Estimation of the Fair Value of Certain Assets and Liabilities of Sun Microsystems, inc. as of January 26, 2010." **[FILED UNDER SEAL]**

11. Attached to this declaration as Exhibit K is a true and correct copy of OAGOOGLE0000062503 - OAGOOGLE0000062726, Oracle Corporation's Form 10-K, filed with the U.S. Securities and Exchange Commission on July 1, 2010.

12. Attached to this declaration as Exhibit L is a true and correct copy of excerpts from OAGOOGLE0000062097, a spreadsheet originally produced in its native format by Oracle. **[FILED UNDER SEAL]**.

13. Attached to this declaration as Exhibit M is a true and correct copy of OAGOOGLE0100071840 - OAGOOGLE0100071986, entitled "SW OEM Pricebook." **[FILED UNDER SEAL]**

14. Attached to this declaration as Exhibit N is a true and correct copy of a January 23, 2001 news press release entitled "Microsoft Reaches Agreement to Settle Contract Dispute With Sun Microsystems," downloaded from http://www.microsoft.com/presspass/press/2001/jan01/01-23sunpr.mspx on June 14, 2011.

15. Attached to this declaration as Exhibit O is a true and correct copy of a Settlement Agreement and Mutual Limited Release downloaded from http://www.microsoft.com/presspass/legal/01-23settlement.mspx on June 14, 2011.

16. Attached to this declaration as Exhibit P is a true and correct copy of OAGOOGLE0100003277 - OAGOOGLE0100003291, a "Stand-Alone TCK License Agreement" entered into by Sun Microsystems, Inc. and Oracle Corporation on March 25, 2004. **[FILED UNDER SEAL]**

17. Attached to this declaration as Exhibit Q is a true and correct copy of OAGOOGLE0100005211 - OAGOOGLE0100005221, a "Stand-Alone TCK License Agreement" entered into by Sun Microsystems, Inc. and SAP AG on May 16, 2005. **[FILED UNDER SEAL]**

18. Attached to this declaration as Exhibit R is a true and correct copy of OAGOOGLE0100165699 - OAGOOGLE0100165746, a July 6, 2010 Oracle presentation entitled "Q1 FY11 Java Sales Review." **[FILED UNDER SEAL]**

19. Attached to this declaration as Exhibit S is a true and correct copy of an Android timeline, downloaded from http://www.android.com/timeline.html on June 14, 2011.

20. Attached to this declaration as Exhibit T is a true and correct copy of http://www.javaworld.com/javaworld/jw-02-2011/110204-android-market.html, downloaded on June 14, 2011.

21. Attached to this declaration as Exhibit U is a true and correct copy of http://www.betanews.com/article/Google-unveils-10-huge-improvements-in-FroYo-Android-22/1274374860, downloaded on June 14, 2011.

22. Attached to this declaration as Exhibit V is a true and correct copy of http://www.tabletsquad.com/top-5-improvements-in-android-3-0/, downloaded f on June 14, 2011.

23. Attached to this declaration as Exhibit W is a true and correct copy of OAGOOGLE0000140115 - OAGOOGLE0000140130, a March 12, 2009 letter from Oracle Corporation Chief Executive Officer Lawrence J. Ellison to the Sun Microsystems, Inc. Board of Directors. **[FILED UNDER SEAL]**

24. Attached to this declaration as Exhibit X is a true and correct copy of http://discussion.forum.nokia.com/forum/showthread.php?11133-j2me-compatibility-between-different-manufacuturers, downloaded f on June 14, 2011.

25. Attached to this declaration as Exhibit Y is a true and correct copy of http://www.russellbeattie.com/blog/1005717, downloaded on June 14, 2011.

26. Attached to this declaration as Exhibit Z is a true and correct copy of http://www.odi.ch/weblog/posting.php?posting=135, downloaded on June 14, 2011.

27. Attached to this declaration as Exhibit AA is a true and correct copy of http://www.oracle.com/technetwork/articles/javame/stateoftheunion-138337.html, downloaded on June 14, 2011.

28. Attached to this declaration as Exhibit BB is a true and correct copy of http://news.cnet.com/8301-13580_3-9800679-39.html?part=rss&subj=news&tag=2547-1_3-0-20, downloaded on June 14, 2011.

29. Attached to this declaration as Exhibit CC is a true and correct copy of http://portal.acm.org/citation.cfm?id=1839348, downloaded on June 14, 2011.

I declare under penalty of perjury that the foregoing facts are true and correct.

Executed on June 14, 2011 in New York, New York.

/s/ Scott T. Weingaertner /s/
Scott T. Weingaertner

Exhibit A

FILED UNDER SEAL

Exhibit B

Oracle v. Google

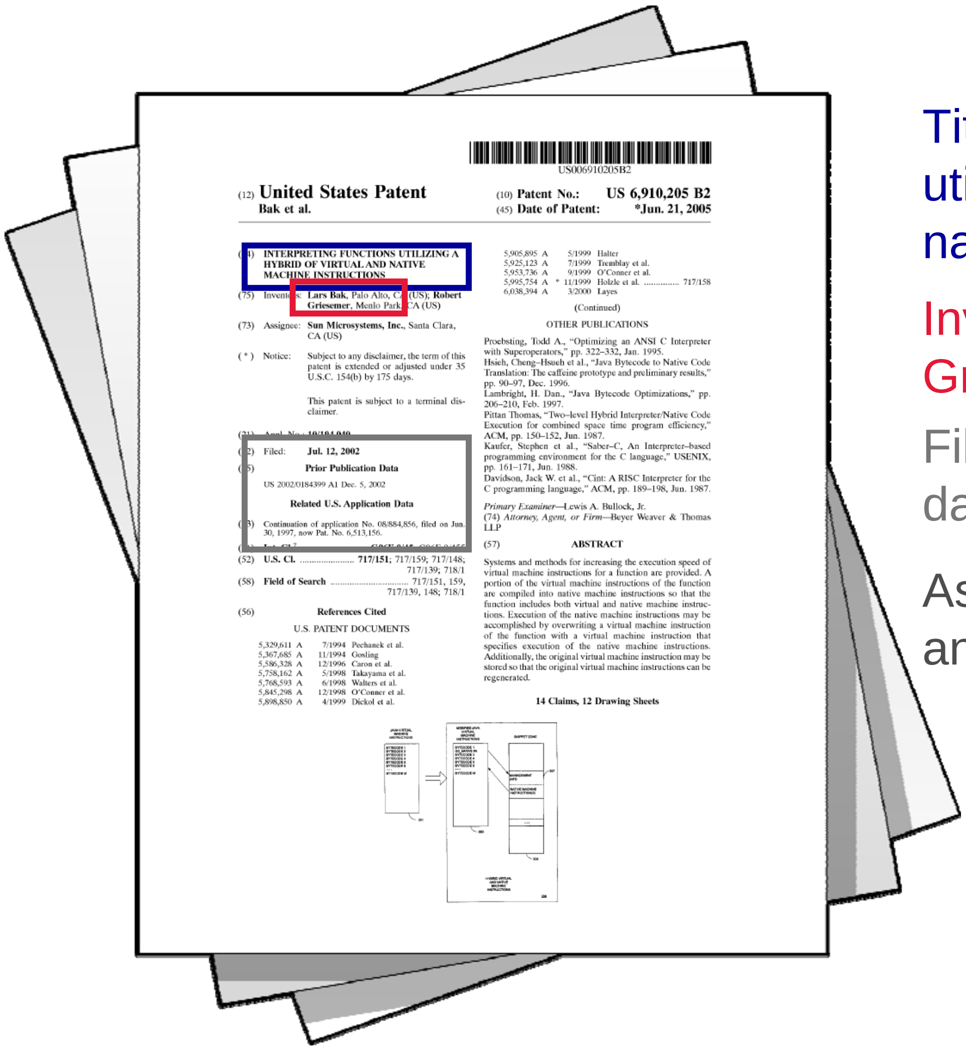
Oracle's Technology Tutorial <u>Supplement</u>

April 6, 2011

Inventions for Performance and Security

- Improved performance
  - RE38,104 (Reference Resolution)
    - "intermediate form [object] code"
    - "resolve" and "resolving"
    - "symbolic [data/field] reference"
  - 6,910,205 (Hybrid Code Execution)
  - 5,966,702 (Class File Redundancy Removal)
    - "reduced class files"
  - 6,061,520 (Play Execution)
    - "play executing step"
  - 7,426,720 (Copy-on-Write Process)
- Improved security
  - 6,125,447 (Fine-Grained Security)
  - 6,192,476 (Call Stack Inspection)

6,910,205 (Hybrid Code Execution)



US006910205B2

(12) **United States Patent**
Bak et al.

(10) **Patent No.:** **US 6,910,205 B2**
(45) **Date of Patent:** ***Jun. 21, 2005**

(54) **INTERPRETING FUNCTIONS UTILIZING A HYBRID OF VIRTUAL AND NATIVE MACHINE INSTRUCTIONS**

(75) Inventors: **Lars Bak**, Palo Alto, CA (US); **Robert Griesemer**, Menlo Park, CA (US)

(73) Assignee: **Sun Microsystems, Inc.**, Santa Clara, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 175 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **10/194,040**

(22) Filed: **Jul. 12, 2002**

(65) **Prior Publication Data**

US 2002/0184399 A1 Dec. 5, 2002

Related U.S. Application Data

(63) Continuation of application No. 08/884,856, filed on Jun. 30, 1997, now Pat. No. 6,513,156.

(51) **Int. Cl.**7 .... **G06F 9/45**; G06F 9/455

(52) **U.S. Cl.** **717/151**; 717/159; 717/148; 717/139; 718/1

(58) **Field of Search** 717/151, 159, 717/139, 148; 718/1

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,329,611	A		7/1994	Pechanek et al.
5,367,685	A		11/1994	Gosling
5,586,328	A		12/1996	Caron et al.
5,758,162	A		5/1998	Takayama et al.
5,768,593	A		6/1998	Walters et al.
5,845,298	A		12/1998	O'Conner et al.
5,898,850	A		4/1999	Dickol et al.
5,905,895	A		5/1999	Halter
5,925,123	A		7/1999	Tremblay et al.
5,953,736	A		9/1999	O'Conner et al.
5,995,754	A	*	11/1999	Holzle et al. 717/158
6,038,394	A		3/2000	Layes

(Continued)

OTHER PUBLICATIONS

Proebsting, Todd A., "Optimizing an ANSI C Interpreter with Superoperators," pp. 322–332, Jan. 1995.
Hsieh, Cheng–Hsueh et al., "Java Bytecode to Native Code Translation: The caffeine prototype and preliminary results," pp. 90–97, Dec. 1996.
Lambright, H. Dan., "Java Bytecode Optimizations," pp. 206–210, Feb. 1997.
Pittan Thomas, "Two–level Hybrid Interpreter/Native Code Execution for combined space time program efficiency," ACM, pp. 150–152, Jun. 1987.
Kaufer, Stephen et al., "Saber–C, An Interpreter–based programming environment for the C language," USENIX, pp. 161–171, Jun. 1988.
Davidson, Jack W. et al., "Cint: A RISC Interpreter for the C programming language," ACM, pp. 189–198, Jun. 1987.

Primary Examiner—Lewis A. Bullock, Jr.
(74) *Attorney, Agent, or Firm*—Beyer Weaver & Thomas LLP

(57) **ABSTRACT**

Systems and methods for increasing the execution speed of virtual machine instructions for a function are provided. A portion of the virtual machine instructions of the function are compiled into native machine instructions so that the function includes both virtual and native machine instructions. Execution of the native machine instructions may be accomplished by overwriting a virtual machine instruction of the function with a virtual machine instruction that specifies execution of the native machine instructions. Additionally, the original virtual machine instruction may be stored so that the original virtual machine instructions can be regenerated.

14 Claims, 12 Drawing Sheets

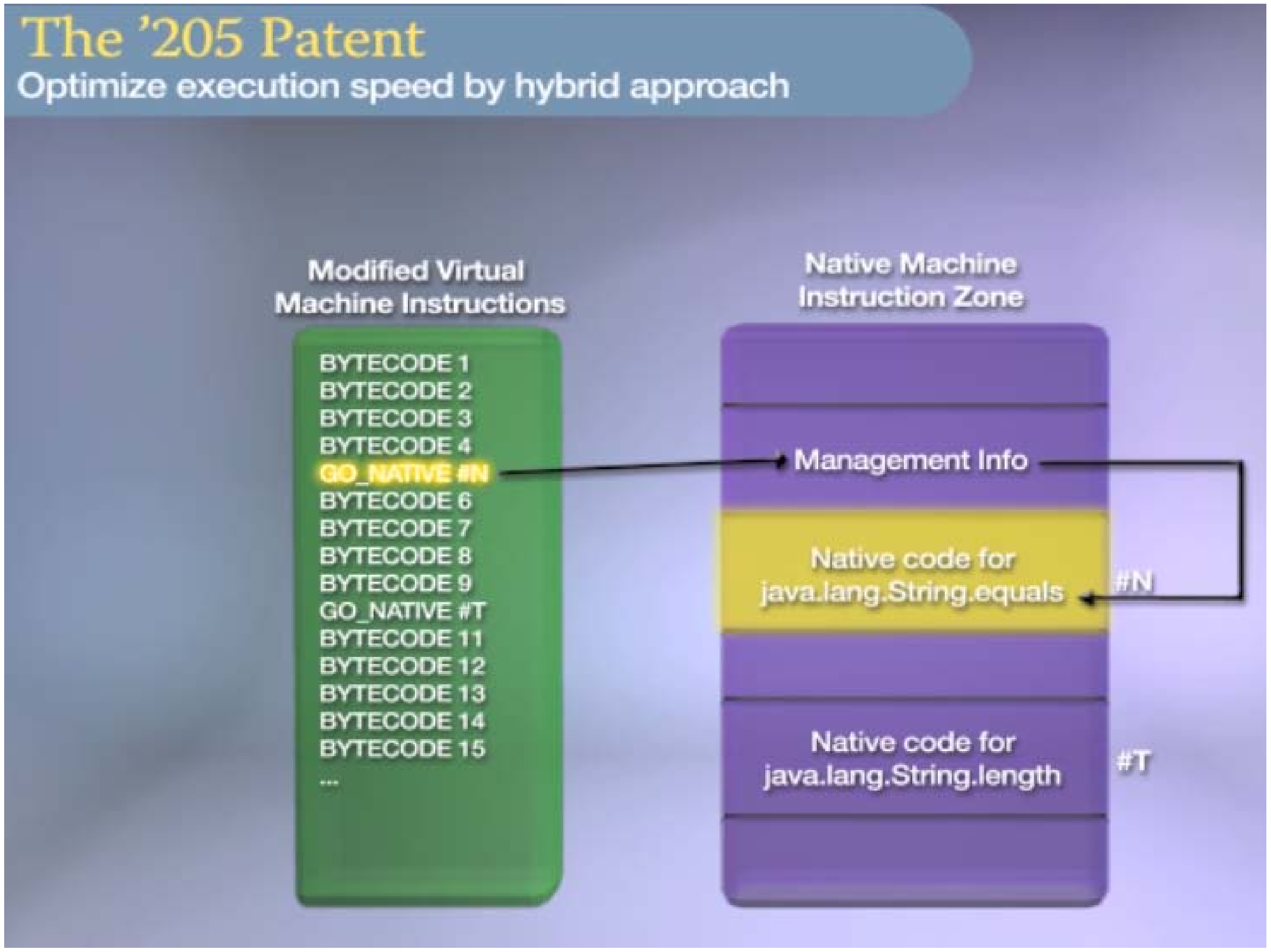
Title: "Interpreting functions utilizing a hybrid of virtual and native machine instructions"

Inventors: Lars Bak, Robert Griesemer

Filed: July 12, 2002 (priority date June 30, 1997)

Asserted Claims: 1, 2, 3, 4, and 8

6,910,205: Hybrid Code Execution



6,910,205: Illustrative Claim

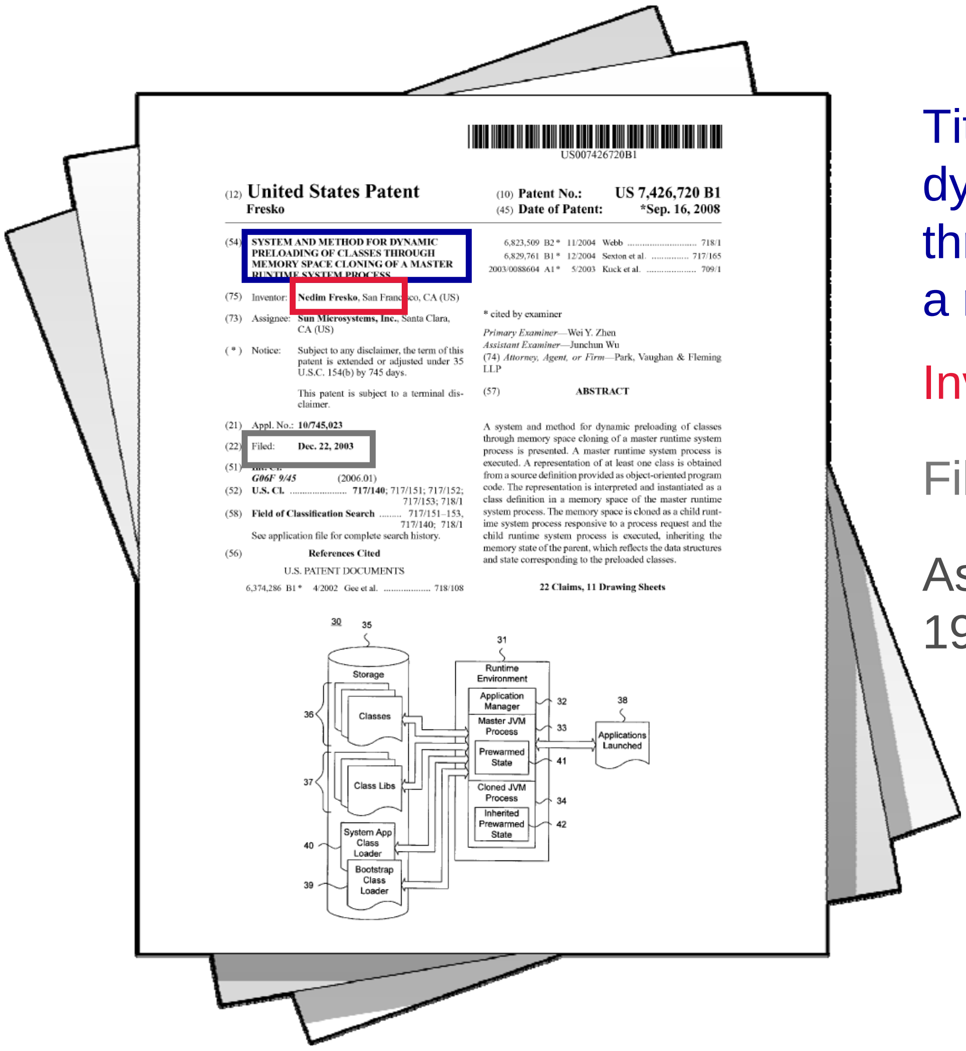
1. In a computer system, a method for increasing the execution speed of virtual machine instructions at runtime, the method comprising:

receiving a first virtual machine instruction;

generating, at runtime, a new virtual machine instruction that represents or references one or more native instructions that can be executed instead of said first virtual machine instruction; and

executing said new virtual machine instruction instead of said first virtual machine instruction.

7,426,720 (Copy-on-Write Process)



US007426720B1

(12) **United States Patent**
Fresko

(10) **Patent No.: US 7,426,720 B1**
(45) **Date of Patent: *Sep. 16, 2008**

(54) **SYSTEM AND METHOD FOR DYNAMIC PRELOADING OF CLASSES THROUGH MEMORY SPACE CLONING OF A MASTER RUNTIME SYSTEM PROCESS**

(75) Inventor: **Nedim Fresko**, San Francisco, CA (US)

(73) Assignee: **Sun Microsystems, Inc.**, Santa Clara, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 745 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **10/745,023**

(22) Filed: **Dec. 22, 2003**

(51) **Int. Cl.**
G06F 9/45 (2006.01)

(52) **U.S. Cl.** **717/140**; 717/151; 717/152; 717/153; 718/1

(58) **Field of Classification Search** 717/151–153, 717/140; 718/1
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,374,286	B1 *	4/2002	Gee et al.	718/108
6,823,509	B2 *	11/2004	Webb	718/1
6,829,761	B1 *	12/2004	Sexton et al.	717/165
2003/0088604	A1 *	5/2003	Kuck et al.	709/1

* cited by examiner

Primary Examiner—Wei Y. Zhen
Assistant Examiner—Junchun Wu
(74) *Attorney, Agent, or Firm*—Park, Vaughan & Fleming LLP

(57) **ABSTRACT**

A system and method for dynamic preloading of classes through memory space cloning of a master runtime system process is presented. A master runtime system process is executed. A representation of at least one class is obtained from a source definition provided as object-oriented program code. The representation is interpreted and instantiated as a class definition in a memory space of the master runtime system process. The memory space is cloned as a child runtime system process responsive to a process request and the child runtime system process is executed, inheriting the memory state of the parent, which reflects the data structures and state corresponding to the preloaded classes.

22 Claims, 11 Drawing Sheets

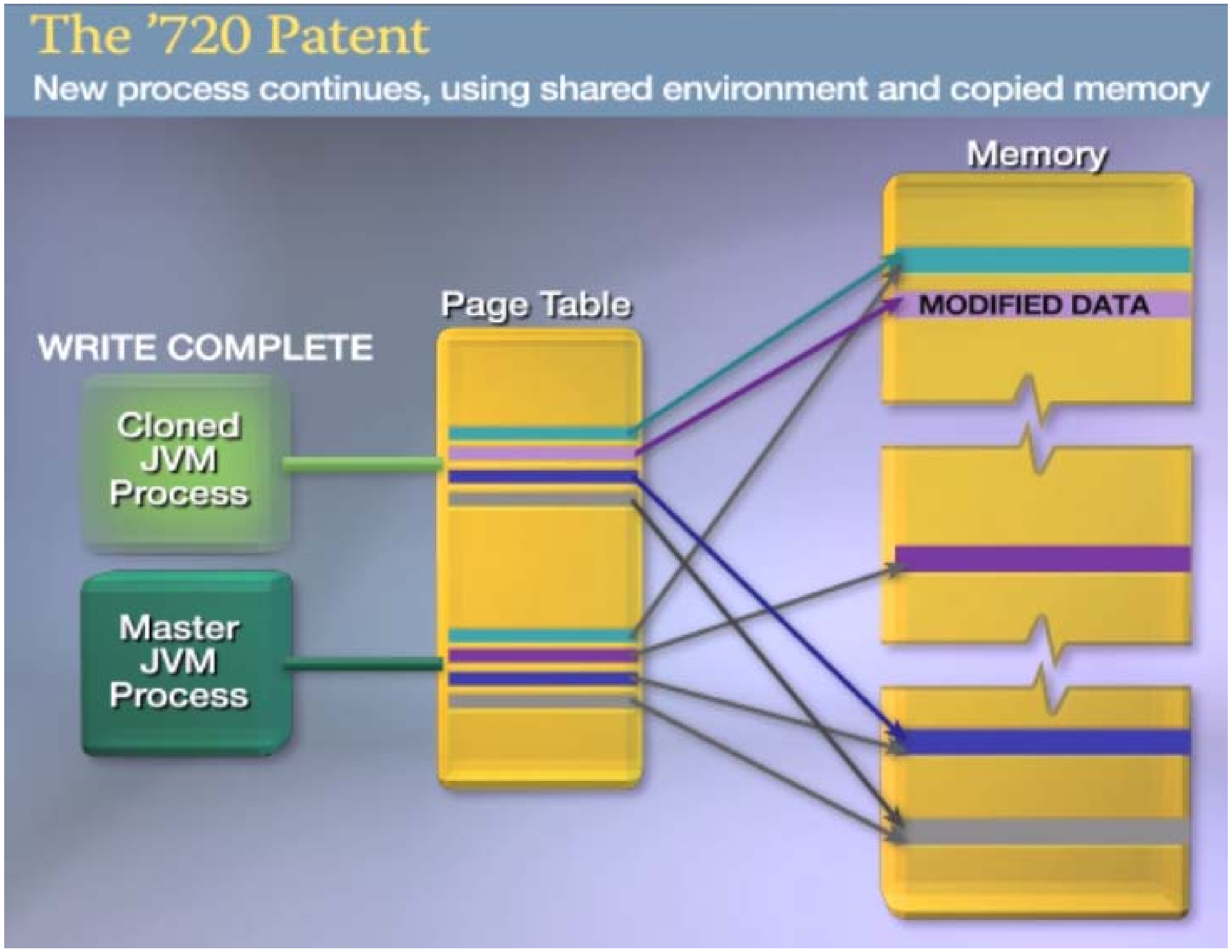
Title: "System and method for dynamic preloading of classes through memory space cloning of a master runtime system process"

Inventor: Nedim Fresko

Filed: December 22, 2003

Asserted Claims: 1-8, 10-17, and 19-22

7,426,720: Copy-on-Write Process



7,426,720: Illustrative Claim

1. A system for dynamic preloading of classes through memory space cloning of a master runtime system process, comprising:

A processor;

A memory

a class preloader to obtain a representation of at least one class from a source definition provided as object-oriented program code;

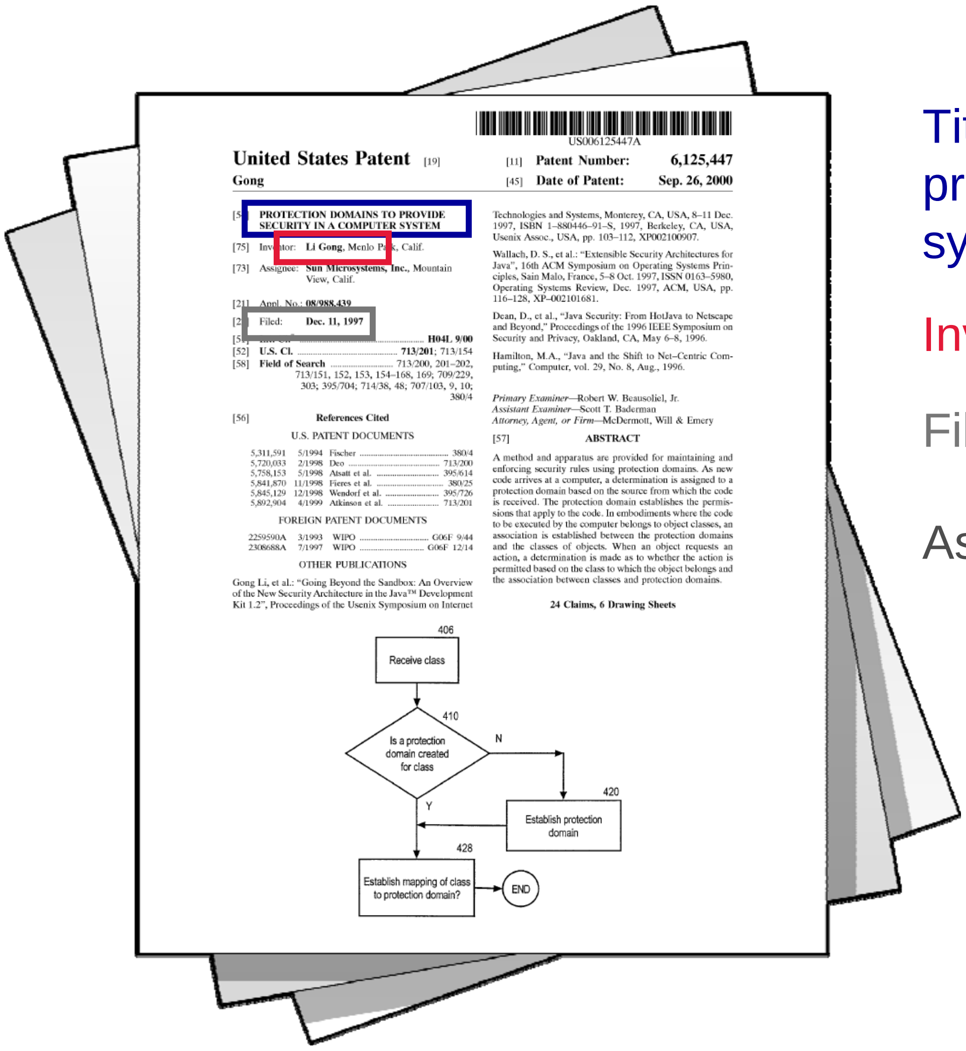
a master runtime system process to interpret and to instantiate the representation as a class definition in a memory space of the master runtime system process;

a runtime environment to clone the memory space as a child runtime system process responsive to a process request and to execute the child runtime system process; and

a copy-on-write process cloning mechanism to instantiate the child runtime system process by copying references to the memory space of the master runtime system process into a separate memory space for the child runtime system process,

and to defer copying of the memory space of the master runtime system process until the child runtime system process needs to modify the referenced memory space of the master runtime system process.

6,125,447 (Fine-Grained Security)



US006125447A

United States Patent [19]
Gong

[11] **Patent Number: 6,125,447**
[45] **Date of Patent: Sep. 26, 2000**

[54] **PROTECTION DOMAINS TO PROVIDE SECURITY IN A COMPUTER SYSTEM**

[75] Inventor: **Li Gong**, Menlo Park, Calif.

[73] Assignee: **Sun Microsystems, Inc.**, Mountain View, Calif.

[21] Appl. No.: **08/988,439**

[22] Filed: **Dec. 11, 1997**

[51] **Int. Cl.** **H04L 9/00**
[52] **U.S. Cl.** **713/201**; 713/154
[58] **Field of Search** 713/200, 201–202, 713/151, 152, 153, 154–168, 169; 709/229, 303; 395/704; 714/38, 48; 707/103, 9, 10; 380/4

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,311,591	5/1994	Fischer	380/4
5,720,033	2/1998	Deo	713/200
5,758,153	5/1998	Atsatt et al.	395/614
5,841,870	11/1998	Fieres et al.	380/25
5,845,129	12/1998	Wendorf et al.	395/726
5,892,904	4/1999	Atkinson et al.	713/201

FOREIGN PATENT DOCUMENTS

2259590A	3/1993	WIPO	G06F 9/44
2308688A	7/1997	WIPO	G06F 12/14

OTHER PUBLICATIONS

Gong Li, et al.: "Going Beyond the Sandbox: An Overview of the New Security Architecture in the Java™ Development Kit 1.2", Proceedings of the Usenix Symposium on Internet Technologies and Systems, Monterey, CA, USA, 8–11 Dec. 1997, ISBN 1–880446–91–S, 1997, Berkeley, CA, USA, Usenix Assoc., USA, pp. 103–112, XP002100907.

Wallach, D. S., et al.: "Extensible Security Architectures for Java", 16th ACM Symposium on Operating Systems Principles, Sain Malo, France, 5–8 Oct. 1997, ISSN 0163–5980, Operating Systems Review, Dec. 1997, ACM, USA, pp. 116–128, XP–002101681.

Dean, D., et al., "Java Security: From HotJava to Netscape and Beyond," Proceedings of the 1996 IEEE Symposium on Security and Privacy, Oakland, CA, May 6–8, 1996.

Hamilton, M.A., "Java and the Shift to Net–Centric Computing," Computer, vol. 29, No. 8, Aug., 1996.

Primary Examiner—Robert W. Beausoliel, Jr.
Assistant Examiner—Scott T. Baderman
Attorney, Agent, or Firm—McDermott, Will & Emery

[57] **ABSTRACT**

A method and apparatus are provided for maintaining and enforcing security rules using protection domains. As new code arrives at a computer, a determination is assigned to a protection domain based on the source from which the code is received. The protection domain establishes the permissions that apply to the code. In embodiments where the code to be executed by the computer belongs to object classes, an association is established between the protection domains and the classes of objects. When an object requests an action, a determination is made as to whether the action is permitted based on the class to which the object belongs and the association between classes and protection domains.

24 Claims, 6 Drawing Sheets

406
Receive class
410
Is a protection domain created for class
N
Y
420
Establish protection domain
428
Establish mapping of class to protection domain?
END

Title: "Protection domains to provide security in a computer system"

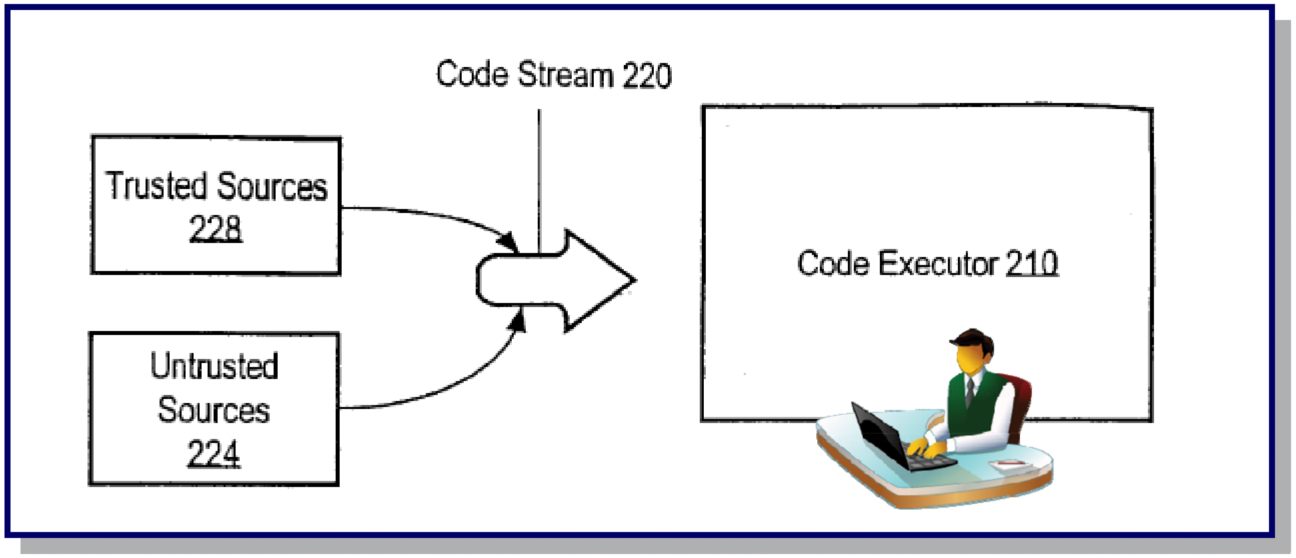
Inventor: Li Gong

Filed: December 11, 1997

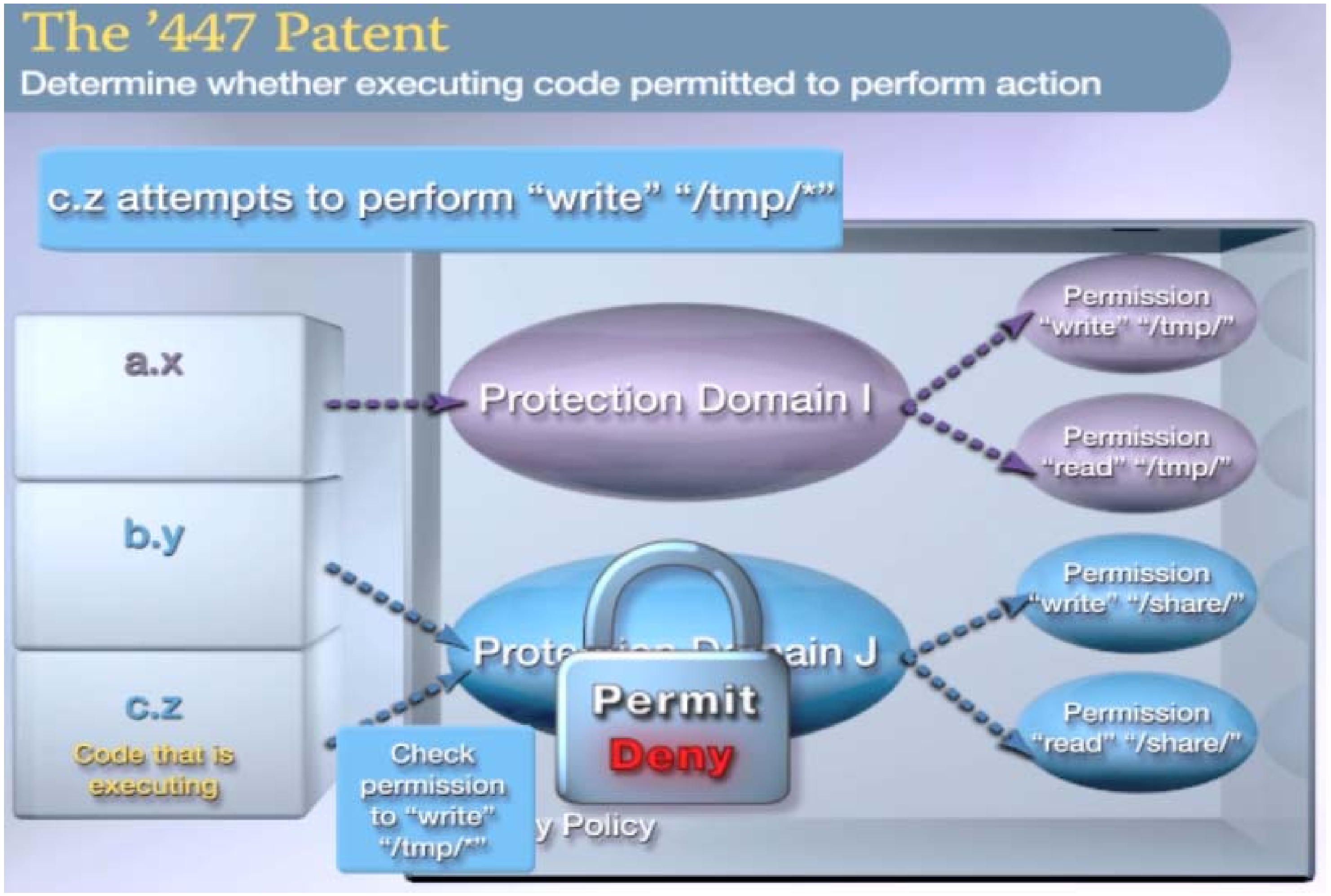
Asserted Claims: 1-24

6,125,447 (Fine-Grained Security)

- End-users download applications from various sources
  - May want to trust applications from certain sources
  - Can't assess whether code is "malicious" (e.g., steal data)
- Executing code may try to perform unauthorized action



6,125,447: Fine-Grained Security


The '447 Patent
Determine whether executing code permitted to perform action
c.z attempts to perform "write" "/tmp/*"
a.x
b.y
c.z
Code that is executing
Protection Domain I
Prote... ...ain J
Permission "write" "/tmp/"
Permission "read" "/tmp/"
Permission "write" "/share/"
Permission "read" "/share/"
Permit
Deny
Check permission to "write" "/tmp/*"
...y Policy

6,125,447: Illustrative Claim

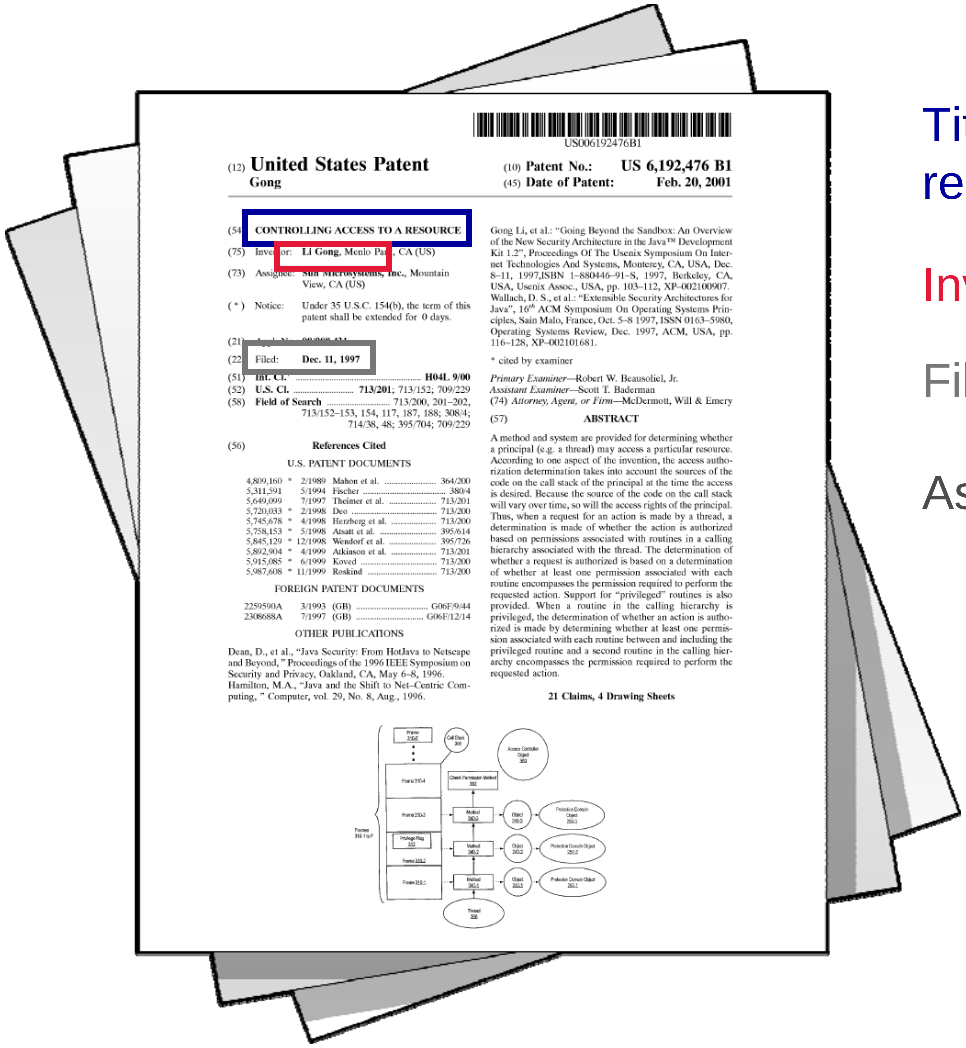
1. A method for providing security, the method comprising the steps of:

establishing one or more protection domains, wherein a protection domain is associated with zero or more permissions;

establishing an association between said one or more protection domains and one or more classes of one or more objects; and

determining whether an action requested by a particular object is permitted based on said association between said one or more protection domains and said one or more classes.

6,192,476 (Call Stack Inspection)



US006192476B1

(12) **United States Patent**
Gong

(10) **Patent No.: US 6,192,476 B1**
(45) **Date of Patent: Feb. 20, 2001**

(54) **CONTROLLING ACCESS TO A RESOURCE**

(75) Inventor: **Li Gong**, Menlo Park, CA (US)

(73) Assignee: **Sun Microsystems, Inc.**, Mountain View, CA (US)

(*) Notice: Under 35 U.S.C. 154(b), the term of this patent shall be extended for 0 days.

(21) Appl. No.: 08/988,431

(22) Filed: **Dec. 11, 1997**

(51) **Int. Cl.**⁷ **H04L 9/00**
(52) **U.S. Cl.** **713/201**; 713/152; 709/229
(58) **Field of Search** 713/200, 201–202, 713/152–153, 154, 117, 187, 188; 308/4; 714/38, 48; 395/704; 709/229

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,809,160	*	2/1989	Mahon et al.	364/200
5,311,591		5/1994	Fischer	380/4
5,649,099		7/1997	Theimer et al.	713/201
5,720,033	*	2/1998	Deo	713/200
5,745,678	*	4/1998	Herzberg et al.	713/200
5,758,153	*	5/1998	Atsatt et al.	395/614
5,845,129	*	12/1998	Wendorf et al.	395/726
5,892,904	*	4/1999	Atkinson et al.	713/201
5,915,085	*	6/1999	Koved	713/200
5,987,608	*	11/1999	Roskind	713/200

FOREIGN PATENT DOCUMENTS

2259590A	3/1993	(GB)	G06F/9/44
2308688A	7/1997	(GB)	G06F/12/14

OTHER PUBLICATIONS

Dean, D., et al., "Java Security: From HotJava to Netscape and Beyond," Proceedings of the 1996 IEEE Symposium on Security and Privacy, Oakland, CA, May 6–8, 1996.
Hamilton, M.A., "Java and the Shift to Net–Centric Computing," Computer, vol. 29, No. 8, Aug., 1996.
Gong Li, et al.: "Going Beyond the Sandbox: An Overview of the New Security Architecture in the Java™ Development Kit 1.2", Proceedings Of The Usenix Symposium On Internet Technologies And Systems, Monterey, CA, USA, Dec. 8–11, 1997,ISBN 1–880446–91–S, 1997, Berkeley, CA, USA, Usenix Assoc., USA, pp. 103–112, XP–002100907.
Wallach, D. S., et al.: "Extensible Security Architectures for Java", 16th ACM Symposium On Operating Systems Principles, Sain Malo, France, Oct. 5–8 1997, ISSN 0163–5980, Operating Systems Review, Dec. 1997, ACM, USA, pp. 116–128, XP–002101681.

* cited by examiner

Primary Examiner—Robert W. Beausoliel, Jr.
Assistant Examiner—Scott T. Baderman
(74) *Attorney, Agent, or Firm*—McDermott, Will & Emery

(57) **ABSTRACT**

A method and system are provided for determining whether a principal (e.g. a thread) may access a particular resource. According to one aspect of the invention, the access authorization determination takes into account the sources of the code on the call stack of the principal at the time the access is desired. Because the source of the code on the call stack will vary over time, so will the access rights of the principal. Thus, when a request for an action is made by a thread, a determination is made of whether the action is authorized based on permissions associated with routines in a calling hierarchy associated with the thread. The determination of whether a request is authorized is based on a determination of whether at least one permission associated with each routine encompasses the permission required to perform the requested action. Support for "privileged" routines is also provided. When a routine in the calling hierarchy is privileged, the determination of whether an action is authorized is made by determining whether at least one permission associated with each routine between and including the privileged routine and a second routine in the calling hierarchy encompasses the permission required to perform the requested action.

21 Claims, 4 Drawing Sheets

Title: "Controlling access to a resource"

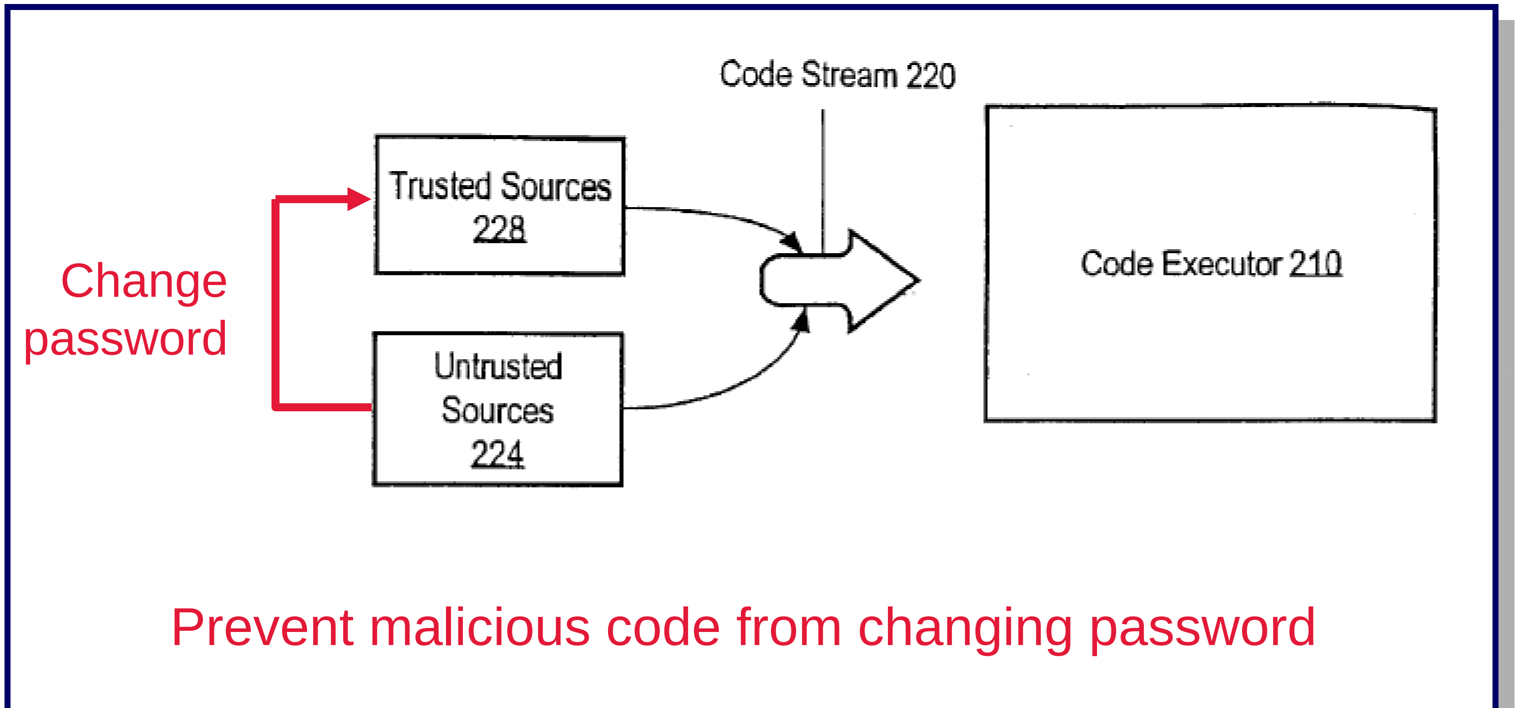
Inventor: Li Gong

Filed: December 11, 1997

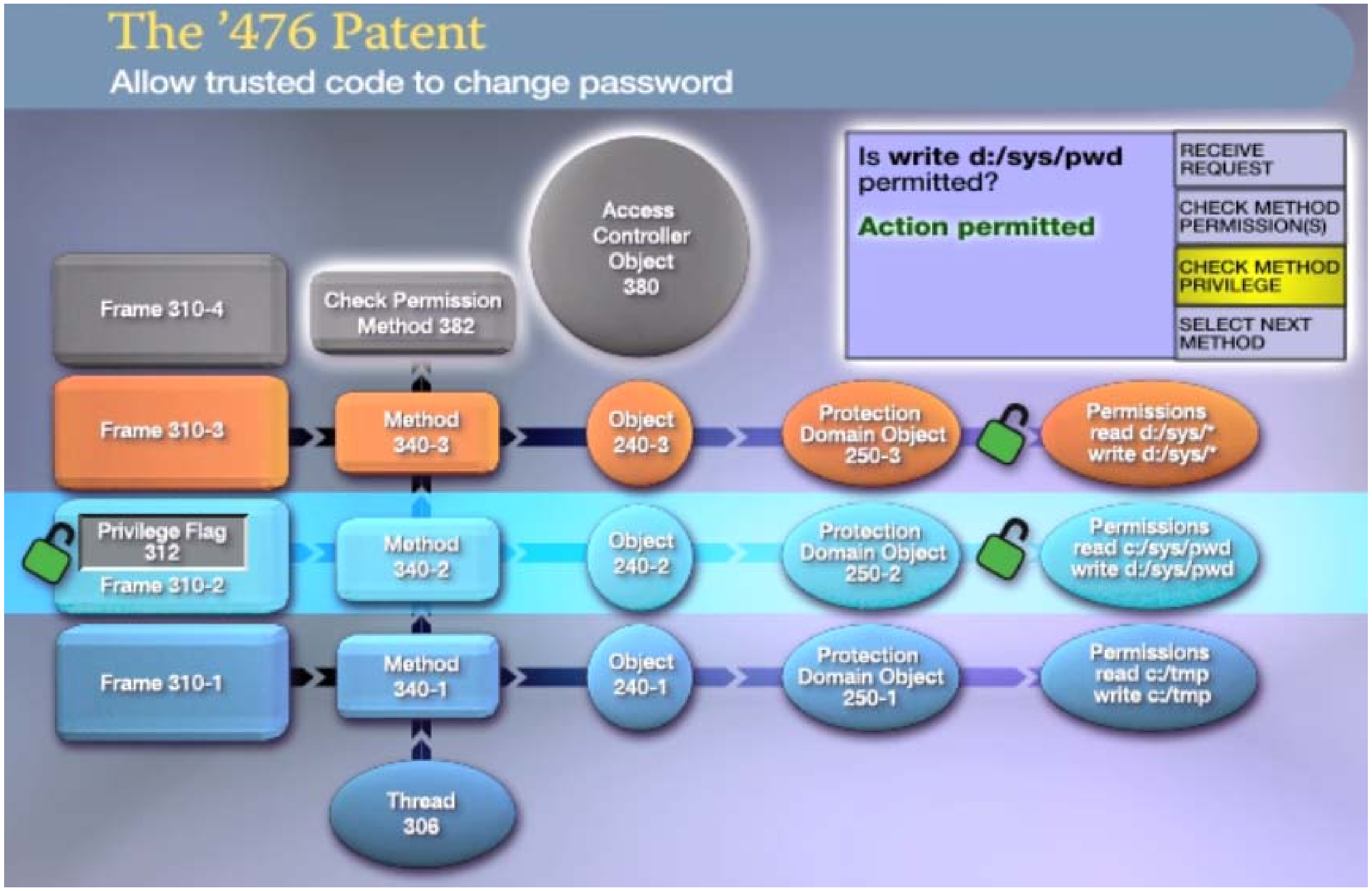
Asserted Claims: 1-21

6,192,476 (Call Stack Inspection)

- Untrusted malicious code may try to invoke trusted code to bypass security protections



6,192,476: Call Stack Inspection


The '476 Patent
Allow trusted code to change password
Is write d:/sys/pwd permitted?
Action permitted
RECEIVE REQUEST
CHECK METHOD PERMISSION(S)
CHECK METHOD PRIVILEGE
SELECT NEXT METHOD
Access Controller Object 380
Check Permission Method 382
Frame 310-4
Frame 310-3
Method 340-3
Object 240-3
Protection Domain Object 250-3
Permissions read d:/sys/* write d:/sys/*
Privilege Flag 312
Frame 310-2
Method 340-2
Object 240-2
Protection Domain Object 250-2
Permissions read c:/sys/pwd write d:/sys/pwd
Frame 310-1
Method 340-1
Object 240-1
Protection Domain Object 250-1
Permissions read c:/tmp write c:/tmp
Thread 306

6,192,476: Illustrative Claim

1. A method for providing security, the method comprising the steps of:

detecting when a request for an action is made by a principal; and

in response to detecting the request, determining whether said action is authorized based on permissions associated with a plurality of routines in a calling hierarchy associated with said principal, wherein said permissions are associated with said plurality of routines based on a first association between protection domains and permissions.

Exhibit C

MORRISON & FOERSTER LLP
MICHAEL A. JACOBS (Bar No. 111664)
mjacobs@mofo.com
MARC DAVID PETERS (Bar No. 211725)
mdpeters@mofo.com
DANIEL P. MUINO (Bar No. 209624)
dmuino@mofo.com
755 Page Mill Road
Palo Alto, CA 94304-1018
Telephone: (650) 813-5600 / Facsimile: (650) 494-0792

BOIES, SCHILLER & FLEXNER LLP
DAVID BOIES (Admitted *Pro Hac Vice*)
dboies@bsfllp.com
333 Main Street
Armonk, NY 10504
Telephone: (914) 749-8200 / Facsimile: (914) 749-8300
STEVEN C. HOLTZMAN (Bar No. 144177)
sholtzman@bsfllp.com
1999 Harrison St., Suite 900
Oakland, CA 94612
Telephone: (510) 874-1000 / Facsimile: (510) 874-1460

ORACLE CORPORATION
DORIAN DALEY (Bar No. 129049)
dorian.daley@oracle.com
DEBORAH K. MILLER (Bar No. 95527)
deborah.miller@oracle.com
MATTHEW M. SARBORARIA (Bar No. 211600)
matthew.sarboraria@oracle.com
500 Oracle Parkway
Redwood City, CA 94065
Telephone: (650) 506-5200 / Facsimile: (650) 506-7114

Attorneys for Plaintiff
ORACLE AMERICA, INC.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN FRANCISCO DIVISION

ORACLE AMERICA, INC. Plaintiff, v. GOOGLE, INC. Defendant.	Case No. 3:10-cv-03561-WHA **ORACLE'S SECOND SUPPLEMENTAL PATENT LOCAL RULE 3-1 DISCLOSURE OF ASSERTED CLAIMS AND INFRINGEMENT CONTENTIONS**

Pursuant to Patent Local Rule 3-1 and agreement between the parties, Plaintiff Oracle America, Inc. ("Oracle") hereby submits the following Second Supplemental Disclosure of Asserted Claims and Infringement Contentions.

Fact discovery is ongoing, and Google has yet to produce substantial quantities of information that may affect Oracle's infringement contentions. In addition, depositions that are directly relevant to Oracle's claims of infringement will be scheduled for after the date of this statement. Not all information about the various versions of the Accused Instrumentalities is publicly available. For example, Google has neither released nor produced the source code for Honeycomb, preventing Oracle from analyzing it. Further still, Oracle understands that Google plans to release future versions of the Accused Instrumentalities.[1]

As such, Oracle's investigation into the extent of infringement by Google is ongoing, and Oracle makes these disclosures based on present knowledge of Google's infringing activities. In light of the foregoing, Oracle reserves the right to supplement or amend these disclosures as further facts are revealed during the course of this litigation.

I. DISCLOSURE OF ASSERTED CLAIMS AND INFRINGEMENT CONTENTIONS.

A. Patent Local Rule 3-1(a) — Asserted Claims.

Oracle asserts that Defendant Google is liable under Title 35 U.S.C. § 271(a), (b), (c), and (f) for infringement of:

- Claims 11-41 of United States Patent No. RE38,104 ("the '104 reissue patent") (infringement claim chart attached as Exhibit A);
- Claims 1, 2, 3, 4, and 8 of United States Patent No. 6,910,205 ("the '205 patent") (infringement claim charts attached as Exhibits B-1 and Exhibit B-2);
- Claims 1, 5-7, 11-13, 15, and 16 of United States Patent No. 5,966,702 ("the '702 patent") (infringement claim chart attached as Exhibit C);

[1] *See, e.g.*, http://en.wikipedia.org/wiki/Android_(operating_system) (last visited March 31, 2011) (Android version "Ice Cream" scheduled for 2011 launch).

- Claims 1-24 of United States Patent No. 6,125,447 ("the '447 patent") (infringement claim chart attached as Exhibit D);
- Claims 1-21 of United States Patent No. 6,192,476 ("the '476 patent") (infringement claim chart attached as Exhibit E);
- Claims 1-4 and 6-23 of United States Patent No. 6,061,520 ("the '520 patent") (infringement claim chart attached as Exhibit F); and
- Claims 1-8, 10-17, and 19-22 of United States Patent No. 7,426,720 ("the '720 patent") (infringement claim chart attached as Exhibit G).

B. Patent Local Rule 3-1(b) — Accused Instrumentalities.

Based on Oracle's investigation thus far, Oracle accuses the following Accused Instrumentalities of infringing the asserted claims specified above in the manner described in Exhibits A-G: (i) "Android" or "the Android Platform";[2] (ii) Google devices running Android; and (iii) other mobile devices running Android. Representative examples of Google devices running Android include the Google Dev Phones, the Google Nexus One, and the Google Nexus S.[3] Representative examples of other mobile devices running Android include HTC's EVO 4G, HTC's Droid Incredible, HTC's G2, Motorola's Droid, and Samsung's Captivate. Android applications, including those written by Google, when built or run will necessarily use the infringing functionality in the manner described in Exhibits A-G. For example, application developers like Google use the Google-provided dx tool from the Android SDK to convert .class

---

[2] "Android" or "the Android Platform" means "Android" as referred to in Google's Answer (Docket No. 32) at Background ¶ 12 and in Google's Answer to Amended Complaint (Docket No. 51) at Background ¶ 12 and at Factual Background ¶¶ 11-17, and includes any versions thereof (whether released or unreleased) and related public or proprietary source code, executable code, and documentation.

[3] *See, e.g.*, JR Raphael, *The Nexus S and Google: Everything There Is To Know*, PCWORLD (Nov. 11, 2010), available at http://www.pcworld.com/article/210460/the_nexus_s_and_google_everything_there_is_to_know.html (last visited Nov. 29, 2010) ("Today's buzz is all about the Samsung Nexus S -- a still-under-wraps smartphone believed to be the successor to Google's Nexus One. According to various leaks, the Nexus S will be a 'Google experience' device, meaning it'll run a stock version of Android without any of those baked-in manufacturer UIs. And, if the latest rumors prove to be true, the Samsung Nexus S will be rocking the as-of-yet-unannounced Android Gingerbread release."). The "leaks" proved to be true: the Nexus S runs a stock version of Gingerbread.

files to a .dex file when building their applications, and thereby infringe the ’520 and ’702 patents. That is the intended use of the dx tool, and there is no substantial non-infringing use of the dx tool.

Google directly infringes the asserted claims enumerated above under 35 U.S.C. § 271(a) because Google, without authority, makes, uses, offers to sell, sells, or imports the Accused Instrumentalities within or into the United States. Further, Google induces the infringement of others under 35 U.S.C. § 271(b) because it contracts with, instructs, and otherwise induces others to make, use, offer to sell, sell, or import the Accused Instrumentalities within or into the United States. Google also contributes to the infringement of others under 35 U.S.C. § 271(c) because it offers to sell, sells, or imports part or all of the Accused Instrumentalities within or into the United States. With respect to the asserted non-method claims of the asserted patents, the Accused Instrumentalities are specially made or adapted for infringement, and are not a staple article suitable for substantial non-infringing use. Further, Google supplies part or all of the Accused Instrumentalities in or from the United States to foreign contractors, including HTC, in violation of 35 U.S.C. § 271(f).

Oracle is not aware of any evidence indicating that anyone, such as a Google partner, OHA member, or downstream licensee, has altered the infringing portions of Google’s Android or Android Platform in any way that is material to the infringement. To the contrary, all available evidence suggests that device manufacturers do not alter the Android operating system in general or the Dalvik virtual machine in particular; and that the changes they do make are generally aimed at the kernel and device drivers (to account for the manufacturer’s particular hardware platform).

The manufacturers’ websites confirm this. Google advertises the Nexus S as “Pure Google” and “The new Android phone from Google.”[4] Samsung states that “Beacuse Nexus S is google experience device, source codes are opened by Google. So, You can find source code for

[4] http://www.google.com/nexus/#/index

the Nexus S at Android Open Source Project site."[5] With respect to Samsung's Captivate, as far as Oracle has been able to determine, for those Android source code files identified in Exhibits A-G that were present in the source code archive for Samsung's Captivate, those files were identical to those from Google's Éclair version of Android.[6] With respect to the source code for the Motorola Droid, Motorola states "All Droid source consists entirely of code found at the Android repo site."[7] With respect to the particular HTC-manufactured devices listed above, the only source code provided by HTC[8] was for the Linux kernel, WebKit and BlueZ, and there was none for Dalvik, the core libraries, or development tools.

Developers have no reason to modify the infringing tools provided by Google for developing Android applications, and Google discourages them from doing so. Google's Android SDK license states:

> 3.3 Except to the extent required by applicable third party licenses, you may not copy (except for backup purposes), modify, adapt, redistribute, decompile, reverse engineer, disassemble, or create derivative works of the SDK or any part of the SDK. Except to the extent required by applicable third party licenses, you may not load any part of the SDK onto a mobile handset or any other hardware device except a personal computer, combine any part of the SDK with other software, or distribute any software or device incorporating a part of the SDK.[9]

Google actively discourages modifications to core Android features through a variety of licensing schemes. For example, Google prohibits anyone from using the Android trademark on a device unless the device is determined to be "Android compatible." Through this requirement, Google ensures that Android devices sold by others will function in the same manner as if they

---

[5] http://opensource.samsung.com/

[6] There was just one exception: the Captivate version of the file *fork.c* in the Linux kernel was identical to the default linux 2.6.29 *fork.c*; there were minor differences with respect to the version of *fork.c* in http://android.git.kernel.org/?p=kernel/linux-2.6.git. These differences had no relation to the infringement by Android that is detailed in Exhibits A-G.

[7] https://opensource.motorola.com/sf/sfmain/do/viewProject/projects.droid

[8] http://developer.htc.com/

[9] http://developer.android.com/sdk/terms.html

were running pure-Google Android, whether or not any modifications were made.[10] Most, perhaps all, of the Accused Instrumentalities bear an Android trademark.

Google makes it clear that there is no need for anyone to modify the infringing code. According to the New York Times just this week, Andy Rubin said that "Android provided the 'basic tools' to allow phone makers to create new models faster, since they did not have to worry about the phone's software. 'They can just focus on innovating a better design,' he said. 'They don't have to worry about adding multitasking and managing memory.'" Jenna Wortham, *Phones Try To Stand Out In a Crowd*, N.Y. TIMES, February 16, 2011. Mr. Rubin is correct that phone makers need not worry about providing multitasking and memory management features, because Google has already provided them in Android. It happens, however, that Google's implementation of these features infringes the '720 patent, among others.

Google's recent actions in the marketplace demonstrate that Android not an open platform but is instead under Google's control. Google has so far refused to release the Honeycomb code as open source. Instead, Google has provided Honeycomb only to its preferred partners, to their mutual advantage, and the disadvantage of everyone else. And according to a recent article, "Google has been demanding that Android licensees abide by 'non-fragmentation clauses' that give Google the final say on how they can tweak the Android code—to make new interfaces and add services—and in some cases whom they can partner with." Ashlee Vance and Peter Burrows, *Do Not Anger the Alpha Android*, BLOOMBERG BUSINESSWEEK, March 30, 2011.

C. Patent Local Rule 3-1(c) — Claim Charts for the Accused Instrumentalities.

Served as Exhibits A-G are claim charts that identify where each element of each asserted claim of the asserted patents is found within the Accused Instrumentalities, based on the information currently available to Oracle.

[10] http://source.android.com/compatibility/android-2.2-cdd.pdf at 8 ("To ensure compatibility with third-party applications, device implementers MUST NOT make any prohibited modifications . . . to these package namespaces: java.*; javax.*; sun.*; android.*; com.android. . . . . Device implementers MAY modify the underlying implementation of the APIs, but such modifications MUST NOT impact the stated behavior and Java-language signature of any publicly exposed APIs.")

The infringement evidence cited in Exhibits A-G is exemplary and not exhaustive. The cited examples are taken from Android 2.2 or 2.3[11] and Google's Android websites. Oracle's infringement contentions apply to all versions of Android having similar or nearly identical code or documentation, including past and expected future releases. Past releases include the Android SDK Preview, 0.9 beta, 1.0, 1.1, 1.5 ("Cupcake"), 1.6 ("Donut"), 2.0/2.1 ("Éclair"), 2.2 ("Froyo"), and 2.3 ("Gingerbread"). Oracle's investigation of "Gingerbread" is ongoing, but Oracle notes that Google has not removed the code Oracle previously identified as infringing.[12]

Although Oracle's investigation is ongoing, the following summary indicates which versions of Android infringe the asserted claims of the specified patents:[13]

- the '104 reissue patent (infringement claim chart previously served as Exhibit A): infringed by all versions of Android subsequent to Oct. 21, 2008, including Android 1.1, 1.5 ("Cupcake"), 1.6 ("Donut"), 2.0/2.1 ("Éclair"), 2.2 ("Froyo"), and 2.3 ("Gingerbread");
- the '205 patent (infringement claim chart previously served as Exhibit B-1): infringed by all versions of Android subsequent to January 28, 2010, including at least Android 2.2 ("Froyo") and 2.3 ("Gingerbread");
- the '205 patent (infringement claim chart previously served as Exhibit B-2): infringed by all versions of Android subsequent to Oct. 21, 2008, including Android 1.1, 1.5 ("Cupcake"), 1.6 ("Donut"), 2.0/2.1 ("Éclair"), 2.2 ("Froyo"), and 2.3 ("Gingerbread");
- the '702 patent (infringement claim chart previously served as Exhibit C): infringed by all versions of Android subsequent to Oct. 21, 2008, including Android 1.1, 1.5

[11] Accessed through http://android.git.kernel.org/ or from Google's production.

[12] Gingerbread continues to not yet be significant in the market when compared to previous versions. As of April 1, 2011, only 2.5% of Android devices checking in with Google were running Gingerbread. http://developer.android.com/resources/dashboard/platform-versions.html (visited Apr. 1, 2011). Most devices are running Froyo or Éclair.

[13] It appears that the Android git source code repository was created on or around Oct. 21, 2008. As such, the following list of infringing Android versions may be expanded based on what Oracle learns about earlier Android versions.

("Cupcake"), 1.6 ("Donut"), 2.0/2.1 ("Éclair"), 2.2 ("Froyo"), and 2.3 ("Gingerbread");

- the '447 patent (infringement claim chart previously served as Exhibit D): infringed by all versions of Android subsequent to Oct. 21, 2008, including Android 1.1, 1.5 ("Cupcake"), 1.6 ("Donut"), 2.0/2.1 ("Éclair"), 2.2 ("Froyo"), and 2.3 ("Gingerbread");
- the '476 patent (infringement claim chart previously served as Exhibit E): infringed by all versions of Android subsequent to Oct. 21, 2008, including Android 1.1, 1.5 ("Cupcake"), 1.6 ("Donut"), 2.0/2.1 ("Éclair"), 2.2 ("Froyo"), and 2.3 ("Gingerbread");
- the '520 patent (infringement claim chart previously served as Exhibit F): infringed by all versions of Android subsequent to Oct. 21, 2008, including Android 1.1, 1.5 ("Cupcake"), 1.6 ("Donut"), 2.0/2.1 ("Éclair"), 2.2 ("Froyo"), and 2.3 ("Gingerbread"); and
- the '720 patent (infringement claim chart previously served as Exhibit G): infringed by all versions of Android subsequent to Oct. 21, 2008, including Android 1.1, 1.5 ("Cupcake"), 1.6 ("Donut"), 2.0/2.1 ("Éclair"), 2.2 ("Froyo"), and 2.3 ("Gingerbread").

D. Patent Local Rule 3-1(d) — Indirect Infringement.

In addition to the acts of direct infringement described above, Google actively contributes to and induces infringement by third parties of each of the asserted claims of the asserted patents. On information and belief, Google purposely and actively distributes the Accused Instrumentalities to manufacturers of products and application developers with the intention that they be used, copied, and distributed to consumers, who in turn use them. Google induces and contributes to the infringement of the asserted claims of each asserted patent, because Google encourages manufacturers, application developers, and service providers (including the members of the Open Handset Alliance), as well as end users, to copy, sell, distribute, re-distribute, and use products that embody or incorporate the Accused Instrumentalities. Google's admissions in its

Amended Counterclaims prove its intent and encouragement of others. (*See, e.g.*, Google's Amended Counterclaims ¶¶ 6-7, 13.) Google contributes to the infringement of others because it offers to sell, sells, or imports part or all of the Accused Instrumentalities within or into the United States. With respect to the asserted non-method claims of the asserted patents, the Accused Instrumentalities are specially made or adapted for infringement, and are not a staple article suitable for substantial non-infringing use.

By providing infringing code and discouraging (and even preventing) modifications, Google further demonstrates the intent necessary for indirect infringement. As discussed below, Google has actual knowledge of Oracle's patents and its infringement is willful.

E. Patent Local Rule 3-1(e) — Nature of Infringement.

Oracle asserts that each element or limitation of each asserted claim of each asserted patent is literally present in the Accused Instrumentalities, except where explicitly indicated. To the extent that any element or limitation of the asserted claims is not found to have literal correspondence in the Accused Instrumentalities, Oracle alleges, on information and belief, that any such elements or limitations are present under the doctrine of equivalents in the Accused Instrumentalities.

F. Patent Local Rule 3.1(f) — Priority Dates.

The '104 reissue patent has a priority date of Dec. 22, 1992, being a continuation of 08/755,764 (filed Nov. 21, 1996) resulting in RE36,204 which is a Reissue of 07/994,655 (filed Dec. 22, 1992) which is U.S. Patent No. 5,367,685.

The '205 patent is a continuation of U.S. Pat. No. 6,513,156, having a priority date of Jun. 30, 1997, the filing date of U.S. patent application number 08/884,856.

G. Patent Local Rule 3.1(g) — Patentee's Asserted Practice of the Claimed Inventions.[14]

1. The '104 Reissue Patent

The following instrumentalities of Oracle practice the asserted claims of the '104 reissue patent:

- JDK 1.0 and subsequent versions;
- JRE 1.1.1 and subsequent versions;
- HotSpot 1.0 and subsequent versions;
- Java SE for Embedded 1.4.2_11 and subsequent versions;
- CDC RI 1.0 and CDC-HI 1.0 and subsequent versions of each;
- CDC AMS 1.0, 1.0_1, 1.0_2, Personal Basis and Personal Profile versions;
- CLDC RI 1.0 and CLDC-HI 1.0 and subsequent versions;
- Foundation Profile 1.0 and subsequent versions;
- J2EE 1.2 (later called Java EE) and subsequent versions;
- WTK 1.0 / Java ME SDK 1.0, and subsequent versions of each;
- Java Real Time 1.0 and all subsequent versions;
- Personal Profile HI and RI 1.0 and subsequent versions;
- Personal Basis Profile-HI and RI 1.0 and subsequent versions;
- PersonalJava 1.0 and subsequent versions;
- EmbeddedJava 1.0 and subsequent versions;
- JavaOS 1.0 (all variants, including Java PC) and subsequent versions;
- Java Card connected platform 3.0 and subsequent versions;
- Oracle Java Wireless Client (formerly Sun Java Wireless Client) 1.0 and subsequent versions;

[14] Oracle's investigation concerning the identification of instrumentalities that practice the asserted claims of the asserted patents is ongoing. There have been many different products relating to the Java Platform over the years, each having many versions or variants, and the lists presented below reflect Oracle's diligent efforts in identifying instrumentalities that practice the asserted claims of the asserted patents.

- MIDP 1.0 and subsequent versions.

2. The ’205 Patent

The following instrumentalities of Oracle practice the asserted claims of the ’205 patent:

- JDK 1.2 and subsequent versions;
- JRE 1.2 and subsequent versions;
- HotSpot 1.0 and subsequent versions;
- Java SE for Embedded 1.4.2 and subsequent versions;
- CDC RI 1.0.1 and CDC-HI 1.0 and subsequent versions of each;
- CDC AMS 1.0, 1.0_1, 1.0_2, Personal Basis and Personal Profile versions;
- CLDC RI 1.1.1;
- CLDC-HI 1.0 and subsequent versions;
- Foundation Profile 1.0.2 and subsequent versions;
- J2EE 1.2 (later called Java EE) and subsequent versions;
- Java ME SDK 3.0 EA and subsequent versions;
- Java Real-Time System 1.0 and all subsequent versions;
- Personal Profile HI and RI 1.0 and subsequent versions; and
- Personal Basis Profile HI and RI 1.0 and subsequent versions.

3. The ’702 Patent

The following instrumentalities of Oracle practice the asserted claims of the ’702 patent:

- PersonalJava (“PJava”) 1.0 and subsequent versions;
- EmbeddedJava (“EJava”) 1.0 and subsequent versions;
- JavaOS 1.0 (and all variants, including Java PC) and subsequent versions;
- CDC RI 1.0 and CDC-HI 1.0, and all subsequent versions of each;
- CDC AMS 1.0, 1.0_1, 1.0_2, Personal Basis and Personal Profile versions;
- CLDC RI 1.1.1 and CLDC-HI 1.0.1, and all subsequent versions of each;
- Personal Profile HI and RI 1.0 and subsequent versions;
- Personal Basis Profile HI and RI 1.0 and subsequent versions;
- Foundation Profile 1.0 and subsequent versions; and

- Java Card platform 2.1 and subsequent versions.

4. The ’447 and ’476 Patents

The following instrumentalities of Oracle practice the asserted claims of the ’447 and ’446 patents:

- JDK 1.2 and subsequent versions;
- JRE 1.2 and subsequent versions;
- Java SE for Embedded 1.4.2_11 and subsequent versions;
- CDC RI 1.0 and CDC-HI 1.0, and all subsequent versions of each;
- CDC AMS 1.0, 1.0_1, 1.0_2, Personal Basis and Personal Profile versions;
- Foundation Profile 1.0.2 and subsequent versions;
- J2EE 1.2 (later called Java EE) and subsequent versions;
- Java ME SDK 3.0 EA and subsequent versions;
- Java Real-Time System 1.0 and all subsequent versions;
- Personal Profile HI and RI 1.0 and subsequent versions;
- Personal Basis Profile HI and RI 1.0 and subsequent versions;
- Java Card connected platform 3.0 and subsequent versions.

Additionally, the following instrumentalities of Oracle practice the asserted claims of the ’447 patent:

- Oracle Java Wireless Client (formerly Sun Java Wireless Client) 1.1.3 and subsequent versions.

5. The ’520 Patent

The following instrumentalities of Oracle practice the asserted claims of the ’520 patent:

- CLDC RI 1.1.1;
- Java Card platform 2.1 and subsequent versions; and
- CLDC-HI 1.1.3 and subsequent versions.

6. The ’720 Patent

The following instrumentalities of Oracle practice the asserted claims of the ’720 patent:

- CDC AMS 1.0, 1.0_1, 1.0_2, Personal Basis and Personal Profile versions.

H. Patent Local Rule 3-1(h) — Willful Infringement.

Google has willfully infringed the patents-in-suit, which are directed to inventions incorporated in the Java Platform. Many factors reveal that Google acted recklessly, *i.e.*, despite a high likelihood that Google's actions infringed a valid and enforceable patent, and that Google actually knew or should have known that its actions constituted an unjustifiably high risk of infringement of a valid and enforceable patent. These factors include:

- Google is a member of the Java Community Process (JCP) and has a seat on the Java SE/EE Executive Committee. *See* Java Community Process homepage, available at http://www.jcp.org/en/participation/committee (last visited Dec. 1, 2010). Through its lengthy participation in the JCP, Google is well aware of the need to obtain a license from Oracle in order to make use of Oracle's Java Platform technologies as Google does in Android. Google's admissions in its Amended Counterclaims prove this awareness. (*See, e.g.*, Google's Amended Counterclaims ¶¶ 6-7, 13.)
- At least three of the seven inventors named in the patents-in-suit, Robert Griesemer, Lars Bak, and Frank Yellin, have left Oracle and work at Google. Their knowledge is attributable to Google.
- Andy Rubin, Google's VP of Mobile Platforms, previously worked at Danger, Inc., which he founded. He understood the need to obtain a license from Oracle (then Sun) to use Java Platform technologies in Danger's Hiptop operating system, and Danger did obtain a commercial license. When Rubin left Danger and founded Android, Inc., he approached Sun about obtaining a commercial license to Java Platform technologies on behalf of Android, Inc. Those discussions ended without Android having obtained a commercial license. Rubin's knowledge is attributable to Google.
- Google has consistently resisted taking a license from Sun for Sun's patented Java Platform technologies.
- In copying Oracle's Java Platform technologies, Google deliberately disregarded a known risk that Oracle had protective patents covering Java Platform technologies.

- Google's Android source code and documentation directly references and copies Java Platform technology specifications, documentation, and source code. *See, e.g.*, mydroid\libcore\security\src\main\java\java\security\CodeSource.java; mydroid\libcore\support\src\test\java\org\apache\harmony\security\tests\support\cert\PolicyNodeImpl.java. Google admits that Android incorporates a subset of Apache Harmony, which it asserts is "an implementation of Sun's Java." (*See, e.g.*, Google's Amended Counterclaims ¶¶ 6-7, 13.)
- Google's website content directly references and demonstrates use of Java Platform technologies. *See, e.g.*, "What is Android?", available at http://developer.android.com/guide/basics/what-is-android.html (last visited Dec. 1, 2010) ("Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language."); Package Index, available at http://developer.android.com/reference/packages.html (last visited Dec. 1, 2010), and subsidiary webpages.
- Google's Android videos directly reference and demonstrate use of Java Platform technologies. *See, e.g.*, Google I/O 2008 Video entitled "Dalvik Virtual Machine Internals," presented by Dan Bornstein (Google), available at http://developer.android.com/videos/index.html#v=ptjedOZEXPM (last visited Dec. 1, 2010).
- As noted above, Google has not removed the code Oracle identified as infringing; Google's direct and indirect infringement is ongoing.

II. DOCUMENT PRODUCTION ACCOMPANYING DISCLOSURES.[15]

A. Patent Local Rule 3-2(a) — Documents Evidencing Pre-Application Disclosure.[16]

Copies of documents produced pursuant to Patent Local Rule 3-2(a) are at OAGOOGLE0000052860-53265, OAGOOGLE0000053266 -53749, OAGOOGLE0000053750-53759, OAGOOGLE0000059578, and OAGOOGLE0000059579-60385. Oracle also directs Google to three public websites: developer.sun.com, java.sun.com, and www.sun.com. Oracle's proprietary commercial releases will be made available for inspection subject to a Protective Order entered in this case or by agreement of the parties.

B. Patent Local Rule 3-2(b) — Documents Evidencing Conception and Reduction to Practice.

Copies of documents evidencing conception, reduction to practice, design and development of the claimed inventions are produced at OAGOOGLE0000000001-52022, OAGOOGLE0000053793-57166, and OAGOOGLE0000059571-59577. Oracle also directs Google to three public websites: developer.sun.com, java.sun.com, and www.sun.com. Oracle's proprietary commercial releases will be made available for inspection subject to a Protective Order entered in this case or by agreement of the parties.

C. Patent Local Rule 3-2(c) — File Histories for the Patents-in-Suit.

Copies of the patent file histories are produced at OAGOOGLE0000052023-52859 and OAGOOGLE0000057167-59570. Certified copies of the patents and file histories are produced at OAGOOGLE0000052023-52169, OAGOOGLE0000052194-52253, OAGOOGLE0000052270-52424, OAGOOGLE0000052602-52859, OAGOOGLE0000102583-105959, and OAGOOGLE0000111357-114304.

---

[15] Oracle will make available source code pursuant to Patent Local Rule 3-2 for inspection by Google in accordance with the protective order. Where different versions of specific Oracle source code do not vary with respect to the claimed inventions in suit (including variants and customized versions for specific customers), Oracle will produce the earliest general version practicing the claimed invention to avoid or minimize any duplicative productions.

[16] As Patent Local Rule 3-2(a) states, Oracle's production of a document as required by the rule shall not constitute an admission that such document evidences or is prior art under 35 U.S.C. § 102.

D. Patent Local Rule 3-2(d) — Ownership of the Patents-in-Suit.

Copies of documents evidencing ownership of the patent rights are produced at OAGOOGLE0000053760-53792 and OAGOOGLE0000056022-56028.

E. Patent Local Rule 3-2(e) — Patentee's Asserted Practice of the Claimed Inventions.

Copies of documents sufficient to show the operation of any aspects or elements of instrumentalities Oracle relies upon as embodying the asserted claims can be found at the following three public websites: developer.sun.com, java.sun.com, and www.sun.com. Oracle's proprietary commercial releases will be made available for inspection subject to the Protective Order entered in this case or by agreement of the parties.

Dated: April 1, 2011

MICHAEL A. JACOBS
MARC DAVID PETERS
MORRISON & FOERSTER LLP

By: /s/ Marc David Peters

Attorneys for Plaintiff
ORACLE AMERICA, INC.

CERTIFICATE OF SERVICE

I declare that I am employed with the law firm of Morrison & Foerster LLP, whose address is 755 Page Mill Road, Palo Alto, California 94304-1018. I am not a party to the within cause, and I am over the age of eighteen years.

I further declare that on April 1, 2011, I served a copy of:

ORACLE'S SECOND SUPPLEMENTAL PATENT LOCAL RULE 3-1 DISCLOSURE OF ASSERTED CLAIMS AND PRELIMINARY INFRINGEMENT CONTENTIONS

[✓] **BY ELECTRONIC SERVICE [Fed. Rule Civ. Proc. rule 5(b)]** by electronically mailing a true and correct copy through Morrison & Foerster LLP's electronic mail system to the e-mail address(es) set forth below, or as stated on the attached service list per agreement in accordance with Federal Rules of Civil Procedure rule 5(b).

Robert F. Perry
Scott T. Weingaertner
Bruce W. Baber
Mark H. Francis
Christopher C. Carnaval
KING & SPALDING LLP
1185 Avenue of the Americas
New York, NY 10036-4003

RPerry@kslaw.com
SWeingaertner@kslaw.com
bbaber@kslaw.com
mfrancis@kslaw.com
ccarnaval@kslaw.com

Fax: 212.556.2222

Timothy T. Scott
Geoffrey M. Ezgar
Leo Spooner III
KING & SPALDING, LLP
333 Twin Dolphin Drive, Suite 400
Redwood Shores, CA 94065

TScott@kslaw.com
GEzgar@kslaw.com
LSpooner@kslaw.com

Fax: 650.590.1900

Donald F. Zimmer, Jr.
Cheryl Z. Sabnis
KING & SPALDING LLP
101 Second Street, Suite 2300
San Francisco, CA 94105

fzimmer@kslaw.com
csabnis@kslaw.com

Fax: 415.318.1300

Steven Snyder
KING & SPALDING LLP
100 N. Tryon Street, Suite 3900
Charlotte, NC 28202

ssnyder@kslaw.com

Fax: 704.503.2622

Renny F. Hwang
GOOGLE INC.
1600 Amphitheatre Parkway
Mountain View, CA 94043

rennyhwang@google.com

Fax: 650.618.1806

Ian C. Ballon
Heather Meeker
GREENBERG TRAURIG LLP
1900 University Avenue, 5th Floor
East Palo Alto, CA 94303

ballon@gtlaw.com
meekerh@gtlaw.com

Fax: 650.328.8508

Joseph R. Wetzel
Dana K. Powers
GREENBERG TRAURIG, LLP
153 Townsend Street, 8th Floor
San Francisco, CA 94107

wetzelj@gtlaw.com
powersdk@gtlaw.com

Fax: 415.707.2010

I declare under penalty of perjury that the foregoing is true and correct.

Executed at Palo Alto, California, this 1st day of April, 2011.

Marc David Peters	/s/ Marc David Peters
(typed)	(signature)

Exhibit D

<u>EXHIBIT D</u>
Supplemental Infringement Contentions for the ’447 Patent

NOTE: The infringement evidence cited below is exemplary and not exhaustive. The cited examples are taken from Android 2.2, 2.3, and Google’s Android websites. Oracle’s infringement contentions apply to all versions of Android having similar or nearly identical code or documentation, including past and expected future releases. Although Oracle’s investigation is ongoing, the ’447 patent is infringed by all versions of Android from Oct. 21, 2008 to the present, including Android 1.1, 1.5 (“Cupcake”), 1.6 (“Donut”), 2.0/2.1 (“Éclair”), 2.2 (“Froyo”), and 2.3 (“Gingerbread”)[1].

The cited source code examples are taken from http://android.git.kernel.org/. The citations are shortened and mirror the file paths shown in http://android.git.kernel.org/. For example, “dalvik\vm\native\InternalNative.c” maps to “[platform/dalvik.git] / vm / native / InternalNative.c” (accessible at http://android.git.kernel.org/?p=platform/dalvik.git;a=blob;f=vm/native/InternalNative.c).

It appears that the Android git source code repository (accessible through http://android.git.kernel.org/) was created on or around Oct. 21, 2008. As such, the list of infringing Android versions may be expanded based on what Oracle learns about earlier Android versions.

Oracle has determined that Android devices execute much of the code cited below when a developer runs the Android Compatibility Test Suite (CTS), which Google requires manufacturers to execute to certify devices as Android-compatible.[2] The mobile device emulator that Google includes with the Android SDK[3] supports Oracle’s conclusion. The emulator displays log messages to inform developers of what is running on the virtual device. If the developer includes a logging command in part of a program, the emulator will output a log entry every time that part of the program is executed. A developer might use this feature, for example, to test whether an application starts to execute a particular section of code before failing. By adding logging commands to key portions of the Android source code cited below, building an Android system image, and loading the code into Google’s emulator, Oracle

[1] Oracle’s investigation into the extent of Gingerbread’s infringement is still ongoing. Gingerbread infringes at least the computer readable medium claims as the code cited in the chart below appears in Gingerbread. For example, the GIT repository, a computer readable medium, is maintained by Google and carries the sequences of instructions listed in the chart below. Oracle continues testing to determine the circumstances under which code from the different versions of Android is executed.

[2] http://source.android.com/compatibility/android-2.2-cdd.pdf at 10 (“To be considered compatible with Android 2.2, device implementations . . . MUST pass the most recent version of the Android Compatibility Test Suite (CTS) available at the time of the device implementation's software is completed.”).

[3] See http://developer.android.com/guide/developing/devices/emulator.html (“The Android SDK includes a virtual mobile device emulator that runs on your computer. The emulator lets you prototype, develop, and test Android applications without using a physical device. The Android emulator mimics all of the hardware and software features of a typical mobile device, except that it cannot place actual phone calls.”).

determined that many of these code sections are executed as part of Google's CTS testing. Thus, Android-compatible devices, when used as Google intends, execute infringing code.

The asserted claims include system, computer-readable medium, and method claims. Anyone who makes, uses, offers to sell, sells, or imports a device running Android within or into the United States directly infringes the system claims. This includes Google and its downstream licensees, including device manufacturers, carriers, application developers, and end users. Similarly, anyone who engages in the above conduct with respect to storage devices containing Android code directly infringes the computer-readable medium claims. This includes Google and its downstream licensees, including device manufacturers and application developers. Anyone who uses a device running Android code directly infringes the method claims. This includes Google and its downstream licensees, including device manufacturers, carriers, application developers, and end users. Google induces and contributes to infringement of all asserted claims by distributing Android code with the intention that it will be executed on mobile devices and by requiring that device manufacturers certify their products by running the CTS as a prerequisite for obtaining access to the Android Market software and branding, among other things. Oracle has confirmed that much of the cited code below is executed when the CTS is run. Google selectively included certain Java APIs in Android while excluding others. The fact that Google selected Java security code for inclusion in Android and has continued to include Java security code in its recent Android releases reflects the functional necessity of this code to the Android platform as a whole. Thus the code cited below is not a staple article suitable for substantial non-infringing use. Google supplies its Android code in and from the United States.

When infringement evidence first presented with respect to one claim is referred to with respect to another, the evidence is applicable because it is not limited to a particular form of infringement.

The '447 Patent	Infringed By
[1-pre] 1. A method for providing security, the method comprising the steps of:	Android includes methods for providing security. *See generally, e.g.*: • dalvik\vm\native\InternalNative.c • dalvik\vm\native\java_security_AccessController.c • dalvik\vm\native\java_lang_VMClassLoader.c • For Froyo: o source code files in dalvik\libcore\security\src\main\java\java\security o source code files in dalvik\libcore\security-kernel\src\main\java\java\security o dalvik\libcore\security\src\main\java\org\apache\harmony\security

	• For Gingerbread o source code files in libcore\luni\src\main\java\java\security o libcore\luni\src\main\java\org\apache\harmony\security *See also, e.g.*: • Android APIs for "java.security," available at http://developer.android.com/reference/java/security/package-summary.html • Android Framework Topics for "Security and Permissions," available at http://developer.android.com/guide/topics/security/security.html • Android Framework Topics for "Security and Permissions" under "The AndroidManifest.xml File," http://developer.android.com/guide/topics/manifest/permission-element.html • Android Framework Topics for "Security and Permissions" under "The AndroidManifest.xml File," http://developer.android.com/guide/topics/manifest/application-element.html • Android Framework Topics for "The AndroidManifest.xml File," available at http://developer.android.com/guide/topics/manifest/manifest-intro.html *See also, e.g.*: • libcore\security\src\test
[1-a] establishing one or more protection domains, wherein a protection domain is associated with zero or more permissions;	Android's security framework establishes one or more protection domains, wherein a protection domain is associated with zero or more permissions. *See, e.g.:* In Froyo, dalvik\libcore\security\src\main\java\java\security\ProtectionDomain.java In Gingerbread, libcore\luni\src\main\java\java\security\ProtectionDomain.java: /** * {@code ProtectionDomain} represents all permissions that are granted to a * specific code source. The {@link ClassLoader} associates each class with the * corresponding {@code ProtectionDomain}, depending on the location and the * certificates (encapsulates in {@link CodeSource}) it loads the code from. * <p> * A class belongs to exactly one protection domain and the protection domain

	` * can not be changed during the lifetime of the class.`
	` */`
	`public class ProtectionDomain {`
	
	`    // CodeSource for this ProtectionDomain`
	`    private CodeSource codeSource;`
	
	`    // Static permissions for this ProtectionDomain`
	`    private PermissionCollection permissions;`
	
	`    // ClassLoader`
	`    private ClassLoader classLoader;`
	
	`    // Set of principals associated with this ProtectionDomain`
	`    private Principal[] principals;`
	
	`    // false if this ProtectionDomain was constructed with static`
	`    // permissions, true otherwise.`
	`    private boolean dynamicPerms;`
	
	`    /**`
	`     * Constructs a new instance of {@code ProtectionDomain} with the specified`
	`     * code source and the specified static permissions.`
	`     * <p>`
	`     * If {@code permissions} is not {@code null}, the {@code permissions}`
	`     * collection is made immutable by calling`
	`     * {@link PermissionCollection#setReadOnly()} and it is considered as`
	`     * granted statically to this {@code ProtectionDomain}.`
	`     * <p>`
	`     * The policy will not be consulted by access checks against this {@code`
	`     * ProtectionDomain}.`
	`     * <p>`
	`     * If {@code permissions} is {@code null}, the method {@link`

	``` * ProtectionDomain#implies(Permission)} always returns {@code false}. * * @param cs * the code source associated with this domain, maybe {@code * null}. * @param permissions * the {@code PermissionCollection} containing all permissions to * be statically granted to this {@code ProtectionDomain}, maybe * {@code null}. */ public ProtectionDomain(CodeSource cs, PermissionCollection permissions) { this.codeSource = cs; if (permissions != null) { permissions.setReadOnly(); } this.permissions = permissions; //this.classLoader = null; //this.principals = null; //dynamicPerms = false; } /** * Constructs a new instance of {@code ProtectionDomain} with the specified * code source, the permissions, the class loader and the principals. * <p> * If {@code permissions} is {@code null}, and access checks are performed * against this protection domain, the permissions defined by the policy are * consulted. If {@code permissions} is not {@code null}, the {@code * permissions} collection is made immutable by calling * {@link PermissionCollection#setReadOnly()}. If access checks are * performed, the policy and the provided permission collection are checked. * <p> * External modifications of the provided {@code principals} array has no```

	` * impact on this {@code ProtectionDomain}.`
	` *`
	` * @param cs`
	` *          the code source associated with this domain, maybe {@code`
	` *          null}.`
	` * @param permissions`
	` *          the permissions associated with this domain, maybe {@code`
	` *          null}.`
	` * @param cl`
	` *          the class loader associated with this domain, maybe {@code`
	` *          null}.`
	` * @param principals`
	` *          the principals associated with this domain, maybe {@code`
	` *          null}.`
	` */`
	`public ProtectionDomain(CodeSource cs, PermissionCollection permissions,`
	`        ClassLoader cl, Principal[] principals) {`
	`    this.codeSource = cs;`
	`    if (permissions != null) {`
	`        permissions.setReadOnly();`
	`    }`
	`    this.permissions = permissions;`
	`    this.classLoader = cl;`
	`    if (principals != null) {`
	`        this.principals = new Principal[principals.length];`
	`        System.arraycopy(principals, 0, this.principals, 0,`
	`                this.principals.length);`
	`    }`
	`    dynamicPerms = true;`
	`}`
	`…`
	`/**`
	` * Returns the static permissions that are granted to this {@code`

	```
	     * ProtectionDomain}.
	     *
	     * @return the static permissions that are granted to this {@code
	     *         ProtectionDomain}, maybe {@code null}.
	     */
	    public final PermissionCollection getPermissions() {
	        return permissions;
	    }
	```
	 See also, e.g.:
	• Android APIs for "java.security," available at http://developer.android.com/reference/java/security/package-summary.html
	• Android Framework Topics for "Security and Permissions," available at http://developer.android.com/guide/topics/security/security.html
	• Android Framework Topics for "Security and Permissions" under "The AndroidManifest.xml File," http://developer.android.com/guide/topics/manifest/permission-element.html
	• Android Framework Topics for "Security and Permissions" under "The AndroidManifest.xml File," http://developer.android.com/guide/topics/manifest/application-element.html
	• Android Framework Topics for "The AndroidManifest.xml File," available at http://developer.android.com/guide/topics/manifest/manifest-intro.html

[1-b] establishing an association between said one or more protection domains and one or more classes of one or more objects; and	Android's security framework establishes an association between said one or more protection domains and one or more classes of one or more objects.
	
	See Claim 1-a, *supra*.
	
	See also, e.g.:
	dalvik\vm\native\dalvik_system_DexFile.c:
	```
	    /*
	     * private static Class defineClass(String name, ClassLoader loader,
	     *     int cookie, ProtectionDomain pd)
	     *
	     * Load a class from a DEX file.  This is roughly equivalent to defineClass()
	```

```
 * in a regular VM -- it's invoked by the class loader to cause the
 * creation of a specific class.  The difference is that the search for and
 * reading of the bytes is done within the VM.
 *
 * The class name is a "binary name", e.g. "java.lang.String".
 *
 * Returns a null pointer with no exception if the class was not found.
 * Throws an exception on other failures.
 */
static void Dalvik_dalvik_system_DexFile_defineClass(const u4* args,
    JValue* pResult)
{
    StringObject* nameObj = (StringObject*) args[0];
    Object* loader = (Object*) args[1];
    int cookie = args[2];
    Object* pd = (Object*) args[3];
    ClassObject* clazz = NULL;
    DexOrJar* pDexOrJar = (DexOrJar*) cookie;
    DvmDex* pDvmDex;
    char* name;
    char* descriptor;

    name = dvmCreateCstrFromString(nameObj);
    descriptor = dvmDotToDescriptor(name);
    LOGV("--- Explicit class load '%s' 0x%08x\n", descriptor, cookie);
    free(name);

    if (!validateCookie(cookie))
        RETURN_VOID();

    if (pDexOrJar->isDex)
        pDvmDex = dvmGetRawDexFileDex(pDexOrJar->pRawDexFile);
    else
```

	```
	    pDvmDex = dvmGetJarFileDex(pDexOrJar->pJarFile);

	/* once we load something, we can't unmap the storage */
	pDexOrJar->okayToFree = false;

	clazz = dvmDefineClass(pDvmDex, descriptor, loader);
	Thread* self = dvmThreadSelf();
	if (dvmCheckException(self)) {
	    /*
	     * If we threw a "class not found" exception, stifle it, since the
	     * contract in the higher method says we simply return null if
	     * the class is not found.
	     */
	    Object* excep = dvmGetException(self);
	    if (strcmp(excep->clazz->descriptor,
	               "Ljava/lang/ClassNotFoundException;") == 0 ||
	        strcmp(excep->clazz->descriptor,
	               "Ljava/lang/NoClassDefFoundError;") == 0)
	    {
	        dvmClearException(self);
	    }
	    clazz = NULL;
	}

	/*
	 * Set the ProtectionDomain -- do we need this to happen before we
	 * link the class and make it available? If so, we need to pass it
	 * through dvmDefineClass (and figure out some other
	 * stuff, like where it comes from for bootstrap classes).
	 */
	if (clazz != NULL) {
	    //LOGI("SETTING pd '%s' to %p\n", clazz->descriptor, pd);
	    dvmSetFieldObject((Object*) clazz, gDvm.offJavaLangClass_pd, pd);
	```

	```
	        }

	        free(descriptor);
	        RETURN_PTR(clazz);
	    }
	```
	 dalvik\vm\native\java_lang_VMClassLoader.c:
	```
	    /*
	     * java.lang.VMClassLoader
	     */
	    …
	    /*
	     * static Class defineClass(ClassLoader cl, String name,
	     *     byte[] data, int offset, int len, ProtectionDomain pd)
	     *     throws ClassFormatError
	     *
	     * Convert an array of bytes to a Class object.
	     */
	    static void Dalvik_java_lang_VMClassLoader_defineClass(const u4* args,
	        JValue* pResult)
	    {
	        Object* loader = (Object*) args[0];
	        StringObject* nameObj = (StringObject*) args[1];
	        const u1* data = (const u1*) args[2];
	        int offset = args[3];
	        int len = args[4];
	        Object* pd = (Object*) args[5];
	        char* name = NULL;

	        name = dvmCreateCstrFromString(nameObj);
	        LOGE("ERROR: defineClass(%p, %s, %p, %d, %d, %p)\n",
	            loader, name, data, offset, len, pd);
	        dvmThrowException("Ljava/lang/UnsupportedOperationException;",
	```

	```
	        "can't load this type of class file");

	    free(name);
	    RETURN_VOID();
	}

	/*
	 * static Class defineClass(ClassLoader cl, byte[] data, int offset,
	 *     int len, ProtectionDomain pd)
	 *     throws ClassFormatError
	 *
	 * Convert an array of bytes to a Class object. Deprecated version of
	 * previous method, lacks name parameter.
	 */
	static void Dalvik_java_lang_VMClassLoader_defineClass2(const u4* args,
	    JValue* pResult)
	{
	    Object* loader = (Object*) args[0];
	    const u1* data = (const u1*) args[1];
	    int offset = args[2];
	    int len = args[3];
	    Object* pd = (Object*) args[4];

	    LOGE("ERROR: defineClass(%p, %p, %d, %d, %p)\n",
	        loader, data, offset, len, pd);
	    dvmThrowException("Ljava/lang/UnsupportedOperationException;",
	        "can't load this type of class file");

	    RETURN_VOID();
	}
	```
[1-c] determining whether an action requested by a	Android's security framework determines whether an action requested by a particular object is permitted based on said association between said one or more protection domains and said one or

particular object is permitted based on said association between said one or more protection domains and said one or more classes.	more classes.
	
	See Claim 1-a and 1-b, *supra*.
	
	See also, e.g.:
	
	In Froyo, dalvik\libcore\security\src\main\java\java\security\ProtectionDomain.java
	In Gingerbread, libcore\luni\src\main\java\java\security\ProtectionDomain.java::
	`/**`
	` * Indicates whether the specified permission is implied by this {@code`
	` * ProtectionDomain}.`
	` * <p>`
	` * If this {@code ProtectionDomain} was constructed with`
	` * {@link #ProtectionDomain(CodeSource, PermissionCollection)}, the`
	` * specified permission is only checked against the permission collection`
	` * provided in the constructor. If {@code null} was provided, {@code false}`
	` * is returned.`
	` * <p>`
	` * If this {@code ProtectionDomain} was constructed with`
	` * {@link #ProtectionDomain(CodeSource, PermissionCollection, ClassLoader, Principal[])}`
	` * , the specified permission is checked against the policy and the`
	` * permission collection provided in the constructor.`
	` *`
	` * @param permission`
	` *            the permission to check against the domain.`
	` * @return {@code true} if the specified {@code permission} is implied by`
	` *         this {@code ProtectionDomain}, {@code false} otherwise.`
	` */`
	`public boolean implies(Permission permission) {`
	`    // First, test with the Policy, as the default Policy.implies()`
	`    // checks for both dynamic and static collections of the`
	`    // ProtectionDomain passed...`

	```
	    if (dynamicPerms
	        && Policy.getAccessiblePolicy().implies(this, permission)) {
	      return true;
	    }

	    // ... and we get here if
	    // either the permissions are static
	    // or Policy.implies() did not check for static permissions
	    // or the permission is not implied
	    return permissions == null ? false : permissions.implies(permission);
	  }
	```
	
	Android APIs for "ProtectionDomain," available at http://developer.android.com/reference/java/security/ProtectionDomain.html:
	
	public **ProtectionDomain** (CodeSource cs, PermissionCollection permissions)
	
	Since: API Level 1
	
	Constructs a new instance of `ProtectionDomain` with the specified code source and the specified static permissions.
	
	If `permissions` is not `null`, the `permissions` collection is made immutable by calling `setReadOnly()` and it is considered as granted statically to this `ProtectionDomain`.
	
	The policy will not be consulted by access checks against this `ProtectionDomain`.
	
	If `permissions` is `null`, the method `implies(Permission)` always returns `false`.
	
	Parameters
	
	cs the code source associated with this domain, maybe `null`.
	
	permissions the `PermissionCollection` containing all permissions to be

	statically granted to this `ProtectionDomain`, maybe `null`. public **ProtectionDomain** (CodeSource cs, PermissionCollection permissions, ClassLoader cl, Principal[] principals) Since: API Level 1 Constructs a new instance of `ProtectionDomain` with the specified code source, the permissions, the class loader and the principals. If `permissions` is `null`, and access checks are performed against this protection domain, the permissions defined by the policy are consulted. If `permissions` is not `null`, the `permissions` collection is made immutable by calling `setReadOnly()`. If access checks are performed, the policy and the provided permission collection are checked. External modifications of the provided `principals` array has no impact on this `ProtectionDomain`. **Parameters** *cs* the code source associated with this domain, maybe `null`. *permissions* the permissions associated with this domain, maybe `null`. *cl* the class loader associated with this domain, maybe `null`. *principals* the principals associated with this domain, maybe `null`. In Froyo, dalvik\libcore\security\src\main\java\java\security\Policy.java In Gingerbread, libcore\luni\src\main\java\java\security\Policy.java: /**

```
     * Indicates whether the specified {@code Permission} is implied by the
     * {@code PermissionCollection} of the specified {@code ProtectionDomain}.
     *
     * @param domain
     *          the {@code ProtectionDomain} for which the permission should
     *          be granted.
     * @param permission
     *          the {@code Permission} for which authorization is to be
     *          verified.
     * @return {@code true} if the {@code Permission} is implied by the {@code
     *       ProtectionDomain}, {@code false} otherwise.
     */
```

In Froyo, dalvik\libcore\security\src\main\java\java\security\Policy.java

```
    public boolean implies(ProtectionDomain domain, Permission permission) {
        if (domain != null) {
            PermissionCollection total = getPermissions(domain);
            PermissionCollection inherent = domain.getPermissions();
            if (total == null) {
                total = inherent;
            } else if (inherent != null) {
                for (Enumeration<Permission> en = inherent.elements(); en.hasMoreElements();) {
                    total.add(en.nextElement());
                }
            }
            if (total != null && total.implies(permission)) {
                return true;
            }
        }
        return false;
    }
```

In Gingerbread, libcore\luni\src\main\java\java\security\Policy.java:

	```
	public boolean implies(ProtectionDomain domain, Permission permission) {
	   return spiImpl == null ? defaultImplies(domain, permission) : spiImpl
	         .engineImplies(domain, permission);
	}

	private boolean defaultImplies(ProtectionDomain domain, Permission permission) {
	   if (domain == null && permission == null) {
	      throw new NullPointerException();
	   }
	   boolean implies = false;
	   if (domain != null) {
	      PermissionCollection total = getPermissions(domain);
	      PermissionCollection inherent = domain.getPermissions();
	      if (inherent != null) {
	         Enumeration<Permission> en = inherent.elements();
	         while (en.hasMoreElements()) {
	            total.add(en.nextElement());
	         }
	      }
	      try {
	         implies = total.implies(permission);
	      } catch (NullPointerException e) {
	         // return false instead of throwing the NullPointerException
	         implies = false;
	      }
	   }
	   return implies;
	}
	```
	In Froyo, dalvik\libcore\luni\src\main\java\java\lang\SecurityManager.java
	In Gingerbread, libcore\luni\src\main\java\java\lang\SecurityManager.java:
	```
	/**
	 * <strong>Warning:</strong> security managers do <strong>not</strong> provide a
	```

```
 * secure environment for executing untrusted code. Untrusted code cannot be
 * safely isolated within the Dalvik VM.
 *
 * <p>Provides security verification facilities for applications. {@code
 * SecurityManager} contains a set of {@code checkXXX} methods which determine
 * if it is safe to perform a specific operation such as establishing network
 * connections, modifying files, and many more. In general, these methods simply
 * return if they allow the application to perform the operation; if an
 * operation is not allowed, then they throw a {@link SecurityException}. The
 * only exception is {@link #checkTopLevelWindow(Object)}, which returns a
 * boolean to indicate permission.
 */
public class SecurityManager {
…
    /**
     * Checks whether the calling thread is allowed to access the resource being
     * guarded by the specified permission object.
     *
     * @param permission
     *            the permission to check.
     * @throws SecurityException
     *             if the requested {@code permission} is denied according to
     *             the current security policy.
     */
    public void checkPermission(Permission permission) {
        try {
            inCheck = true;
            AccessController.checkPermission(permission);
        } finally {
            inCheck = false;
        }
    }
```

	```
	        /**
	         * Checks whether the specified security context is allowed to access the
	         * resource being guarded by the specified permission object.
	         *
	         * @param permission
	         *            the permission to check.
	         * @param context
	         *            the security context for which to check permission.
	         * @throws SecurityException
	         *             if {@code context} is not an instance of {@code
	         *             AccessControlContext} or if the requested {@code permission}
	         *             is denied for {@code context} according to the current
	         *             security policy.
	         */
	        public void checkPermission(Permission permission, Object context) {
	            try {
	                inCheck = true;
	                // Must be an AccessControlContext. If we don't check
	                // this, then applications could pass in an arbitrary
	                // object which circumvents the security check.
	                if (context instanceof AccessControlContext) {
	                    ((AccessControlContext) context).checkPermission(permission);
	                } else {
	                    throw new SecurityException();
	                }
	            } finally {
	                inCheck = false;
	            }
	        }
	    }
	```
	
	In Froyo, dalvik\libcore\security-kernel\src\main\java\java\security\AccessController.java
	In Gingerbread, libcore\luni\src\main\java\java\security\AccessController.java:

	```
	/**
	 * Checks the specified permission against the VM's current security policy.
	 * The check is performed in the context of the current thread. This method
	 * returns silently if the permission is granted, otherwise an {@code
	 * AccessControlException} is thrown.
	 * <p>
	 * A permission is considered granted if every {@link ProtectionDomain} in
	 * the current execution context has been granted the specified permission.
	 * If privileged operations are on the execution context, only the {@code
	 * ProtectionDomain}s from the last privileged operation are taken into
	 * account.
	 * <p>
	 * This method delegates the permission check to
	 * {@link AccessControlContext#checkPermission(Permission)} on the current
	 * callers' context obtained by {@link #getContext()}.
	 *
	 * @param permission
	 *            the permission to check against the policy
	 * @throws AccessControlException
	 *             if the specified permission is not granted
	 * @throws NullPointerException
	 *             if the specified permission is {@code null}
	 * @see AccessControlContext#checkPermission(Permission)
	 *
	 */
	public static void checkPermission(Permission permission)
	        throws AccessControlException {
	    if (permission == null) {
	        throw new NullPointerException("permission == null");
	    }

	    getContext().checkPermission(permission);
	}
	```

	In Froyo, dalvik\libcore\security-kernel\src\main\java\java\security\AccessControlContext.java
	In Gingerbread, libcore\luni\src\main\java\java\security\AccessControllerContext.java:
	`// List of ProtectionDomains wrapped by the AccessControlContext`
	`// It has the following characteristics:`
	`//     - 'context' can not be null`
	`//     - never contains null(s)`
	`//     - all elements are unique (no dups)`
	`ProtectionDomain[] context;`
	`…`
	`   /**`
	`    * Checks the specified permission against the vm's current security policy.`
	`    * The check is based on this {@code AccessControlContext} as opposed to the`
	`    * {@link AccessController#checkPermission(Permission)} method which`
	`    * performs access checks based on the context of the current thread. This`
	`    * method returns silently if the permission is granted, otherwise an`
	`    * {@code AccessControlException} is thrown.`
	`    * <p>`
	`    * A permission is considered granted if every {@link ProtectionDomain} in`
	`    * this context has been granted the specified permission.`
	`    * <p>`
	`    * If privileged operations are on the call stack, only the {@code`
	`    * ProtectionDomain}s from the last privileged operation are taken into`
	`    * account.`
	`    * <p>`
	`    * If inherited methods are on the call stack, the protection domains of the`
	`    * declaring classes are checked, not the protection domains of the classes`
	`    * on which the method is invoked.`
	`    *`
	`    * @param perm`
	`    *            the permission to check against the policy`
	`    * @throws AccessControlException`
	`    *             if the specified permission is not granted`

	```
	  * @throws NullPointerException
	  *         if the specified permission is {@code null}
	  * @see AccessController#checkPermission(Permission)
	  */
	 public void checkPermission(Permission perm) throws AccessControlException {
	    if (perm == null) {
	       throw new NullPointerException("Permission cannot be null");
	    }
	    for (int i = 0; i < context.length; i++) {
	       if (!context[i].implies(perm)) {
	          throw new AccessControlException("Permission check failed "
	                + perm, perm);
	       }
	    }
	    if (inherited != null) {
	       inherited.checkPermission(perm);
	    }
	 }
	```

The ’447 Patent	Infringed By
2. The method of claim 1, wherein:	*See* Claim 1, *supra*.

at least one protection domain of said one or more protection domains is associated with a code identifier;	*See* Claim 1-a and 1-b, *supra*.
	
	E.g.:
	dalvik\vm\native\dalvik_system_DexFile.c:
	```
	         /*
	          * private static Class defineClass(String name, ClassLoader loader,
	          *     int cookie, ProtectionDomain pd)
	          *
	          * Load a class from a DEX file.  This is roughly equivalent to defineClass()
	          * in a regular VM -- it's invoked by the class loader to cause the
	```

The ’447 Patent	Infringed By

	` * creation of a specific class.  The difference is that the search for and`
	` * reading of the bytes is done within the VM.`
	` *`
	` * The class name is a "binary name", e.g. "java.lang.String".`
	` *`
	` * Returns a null pointer with no exception if the class was not found.`
	` * Throws an exception on other failures.`
	` */`
	`static void Dalvik_dalvik_system_DexFile_defineClass(const u4* args,`
	`    JValue* pResult)`
	`{`
	`    StringObject* nameObj = (StringObject*) args[0];`
	`    Object* loader = (Object*) args[1];`
	`    int cookie = args[2];`
	`    Object* pd = (Object*) args[3];`
	`    ClassObject* clazz = NULL;`
	`    DexOrJar* pDexOrJar = (DexOrJar*) cookie;`
	`    DvmDex* pDvmDex;`
	`    char* name;`
	`    char* descriptor;`
	
	`    name = dvmCreateCstrFromString(nameObj);`
	`    descriptor = dvmDotToDescriptor(name);`
	`    LOGV("--- Explicit class load '%s' 0x%08x\n", descriptor, cookie);`
	`    free(name);`
	
	`    if (!validateCookie(cookie))`
	`        RETURN_VOID();`
	
	`    if (pDexOrJar->isDex)`
	`        pDvmDex = dvmGetRawDexFileDex(pDexOrJar->pRawDexFile);`
	`    else`

The ’447 Patent	Infringed By
	```
	        pDvmDex = dvmGetJarFileDex(pDexOrJar->pJarFile);

	    /* once we load something, we can't unmap the storage */
	    pDexOrJar->okayToFree = false;

	    clazz = dvmDefineClass(pDvmDex, descriptor, loader);
	    Thread* self = dvmThreadSelf();
	    if (dvmCheckException(self)) {
	        /*
	         * If we threw a "class not found" exception, stifle it, since the
	         * contract in the higher method says we simply return null if
	         * the class is not found.
	         */
	        Object* excep = dvmGetException(self);
	        if (strcmp(excep->clazz->descriptor,
	                   "Ljava/lang/ClassNotFoundException;") == 0 ||
	            strcmp(excep->clazz->descriptor,
	                   "Ljava/lang/NoClassDefFoundError;") == 0)
	        {
	            dvmClearException(self);
	        }
	        clazz = NULL;
	    }

	    /*
	     * Set the ProtectionDomain -- do we need this to happen before we
	     * link the class and make it available? If so, we need to pass it
	     * through dvmDefineClass (and figure out some other
	     * stuff, like where it comes from for bootstrap classes).
	     */
	    if (clazz != NULL) {
	        //LOGI("SETTING pd '%s' to %p\n", clazz->descriptor, pd);
	```

The ’447 Patent	Infringed By

	`            dvmSetFieldObject((Object*) clazz, gDvm.offJavaLangClass_pd, pd);`
	`        }`
	
	`        free(descriptor);`
	`        RETURN_PTR(clazz);`
	`    }`
	
	E.g.:
	dalvik\vm\native\java_lang_VMClassLoader.c:
	`    /*`
	`     * java.lang.VMClassLoader`
	`     */`
	`    …`
	`    /*`
	`     * static Class defineClass(ClassLoader cl, String name,`
	`     *     byte[] data, int offset, int len, ProtectionDomain pd)`
	`     *     throws ClassFormatError`
	`     *`
	`     * Convert an array of bytes to a Class object.`
	`     */`
	`    static void Dalvik_java_lang_VMClassLoader_defineClass(const u4* args,`
	`        JValue* pResult)`
	`    {`
	`        Object* loader = (Object*) args[0];`
	`        StringObject* nameObj = (StringObject*) args[1];`
	`        const u1* data = (const u1*) args[2];`
	`        int offset = args[3];`
	`        int len = args[4];`
	`        Object* pd = (Object*) args[5];`
	`        char* name = NULL;`
	
	`        name = dvmCreateCstrFromString(nameObj);`

The ’447 Patent	Infringed By
	```
	    LOGE("ERROR: defineClass(%p, %s, %p, %d, %d, %p)\n",
	        loader, name, data, offset, len, pd);
	    dvmThrowException("Ljava/lang/UnsupportedOperationException;",
	        "can't load this type of class file");

	    free(name);
	    RETURN_VOID();
	}

	/*
	 * static Class defineClass(ClassLoader cl, byte[] data, int offset,
	 *     int len, ProtectionDomain pd)
	 *     throws ClassFormatError
	 *
	 * Convert an array of bytes to a Class object. Deprecated version of
	 * previous method, lacks name parameter.
	 */
	static void Dalvik_java_lang_VMClassLoader_defineClass2(const u4* args,
	    JValue* pResult)
	{
	    Object* loader = (Object*) args[0];
	    const u1* data = (const u1*) args[1];
	    int offset = args[2];
	    int len = args[3];
	    Object* pd = (Object*) args[4];

	    LOGE("ERROR: defineClass(%p, %p, %d, %d, %p)\n",
	        loader, data, offset, len, pd);
	    dvmThrowException("Ljava/lang/UnsupportedOperationException;",
	        "can't load this type of class file");

	    RETURN_VOID();
	```

The '447 Patent	Infringed By

	}
	
	 See also, e.g.:
	
	 In Froyo, dalvik\libcore\security\src\main\java\java\security\CodeSource.java
	 In Gingerbread, libcore\luni\src\main\java\java\security\CodeSource.java:
	 /**
	  * {@code CodeSource} encapsulates the location from where code is loaded and
	  * the certificates that were used to verify that code. This information is used
	  * by {@code SecureClassLoader} to define protection domains for loaded classes.
	  *
	  * @see SecureClassLoader
	  * @see ProtectionDomain
	  */
	 public class CodeSource implements Serializable {
	
	     private static final long serialVersionUID = 4977541819976013951L;
	
	     // Location of this CodeSource object
	     private URL location;
	
	     // Array of certificates assigned to this CodeSource object
	     private transient java.security.cert.Certificate[] certs;
	
	     // Array of CodeSigners
	     private transient CodeSigner[] signers;
	
	     // SocketPermission() in implies() method takes to many time.
	     // Need to cache it for better performance.
	     private transient SocketPermission sp;
	
	     // Cached factory used to build CertPath-s in <code>getCodeSigners()</code>.

The ’447 Patent	Infringed By

	private transient CertificateFactory factory;
	
	/**
	 * Constructs a new instance of {@code CodeSource} with the specified
	 * {@code URL} and the {@code Certificate}s.
	 *
	 * @param location
	 * the {@code URL} representing the location from where code is
	 * loaded, maybe {@code null}.
	 * @param certs
	 * the {@code Certificate} used to verify the code, loaded from
	 * the specified {@code location}, maybe {@code null}.
	 */
	public CodeSource(URL location, Certificate[] certs) {
	    this.location = location;
	    if (certs != null) {
	        this.certs = new Certificate[certs.length];
	        System.arraycopy(certs, 0, this.certs, 0, certs.length);
	    }
	}
	
	/**
	 * Constructs a new instance of {@code CodeSource} with the specified
	 * {@code URL} and the {@code CodeSigner}s.
	 *
	 * @param location
	 * the {@code URL} representing the location from where code is
	 * loaded, maybe {@code null}.
	 * @param signers
	 * the {@code CodeSigner}s of the code, loaded from the specified
	 * {@code location}. Maybe {@code null}.
	 */

The ’447 Patent	Infringed By
	```
	public CodeSource(URL location, CodeSigner[] signers) {
	    this.location = location;
	    if (signers != null) {
	        this.signers = new CodeSigner[signers.length];
	        System.arraycopy(signers, 0, this.signers, 0, signers.length);
	    }
	}
	…
	```
at least one class of said one or more classes is associated with said code identifier; and	*See* Claim 1-b, *supra*, and above.
the step of establishing an association between said one or more protection domains and said one or more classes of one or more objects further includes the step of associating said one or more protection domains and said one or more classes based on said code identifier.	*See* Claim 1, *supra*, and above.

The ’447 Patent	Infringed By
3. The method of claim 2, wherein said code identifier indicates a source of code used to define each class of said one or more classes.	*See* Claim 2, *supra*.

The ’447 Patent	Infringed By
4. The method of claim 2, wherein said code identifier indicates a key	*See* Claim 2, *supra*.

The ’447 Patent	Infringed By

associated with each class of said one or more classes.	The certificate mentioned in Claim 2, *supra*, includes a key.
	
	See, e.g.:
	
	In Froyo, dalvik\libcore\security\src\main\java\java\security\CodeSource.java
	In Gingerbread, libcore\luni\src\main\java\java\security\CodeSource.java:
	`/**`
	` * {@code CodeSource} encapsulates the location from where code is loaded and`
	` * the certificates that were used to verify that code. This information is used`
	` * by {@code SecureClassLoader} to define protection domains for loaded classes.`
	` *`
	` * @see SecureClassLoader`
	` * @see ProtectionDomain`
	` */`
	`…`
	`    // Array of certificates assigned to this CodeSource object`
	`    private transient java.security.cert.Certificate[] certs;`
	
	`…`
	`    /**`
	`     * Constructs a new instance of {@code CodeSource} with the specified`
	`     * {@code URL} and the {@code Certificate}s.`
	`     *`
	`     * @param location`
	`     *            the {@code URL} representing the location from where code is`
	`     *            loaded, maybe {@code null}.`
	`     * @param certs`
	`     *            the {@code Certificate} used to verify the code, loaded from`
	`     *            the specified {@code location}, maybe {@code null}.`
	`     */`
	`    public CodeSource(URL location, Certificate[] certs) {`
	`        this.location = location;`

The '447 Patent	Infringed By

	```
	        if (certs != null) {
	            this.certs = new Certificate[certs.length];
	            System.arraycopy(certs, 0, this.certs, 0, certs.length);
	        }
	    }
	…
	    /**
	     * Returns the certificates of this {@code CodeSource}. If the
	     * {@link #CodeSource(URL, CodeSigner[])} constructor was used to create
	     * this instance, the certificates are obtained from the supplied signers.
	     * <p>
	     * External modifications of the returned {@code Certificate[]} has no
	     * impact on this {@code CodeSource}.
	     *
	     * @return the certificates of this {@code CodeSource} or {@code null} if
	     *         there is none.
	     */
	    public final Certificate[] getCertificates() {
	        getCertificatesNoClone();
	        if (certs == null) {
	            return null;
	        }
	        Certificate[] tmp = new Certificate[certs.length];
	        System.arraycopy(certs, 0, tmp, 0, certs.length);
	        return tmp;
	    }
	…
	```
	In Froyo, dalvik\libcore\security\src\main\java\java\security\Certificate.java
	In Gingerbread, libcore\luni\src\main\java\java\security\Certificate.java:
	```
	/**
	 * {@code Certificate} represents an identity certificate, such as X.509 or PGP.
	```

The '447 Patent	Infringed By
	* Note: A {@code Certificate} instances does not make any statement about the * validity of itself. It's in the responsibility of the application to verify * the validity of its certificates. * * @deprecated Replaced by behavior in {@link java.security.cert} * @see java.security.cert.Certificate */ X.509 is an internet standard certificate format. *See, e.g.*, RFC2459, available at www.ietf.org/rfc/rfc2459.txt (discussing keys and certificates). Information about PGP certificates is available at, *e.g.*, www.pgpi.org; http://en.wikipedia.org/wiki/Pretty_Good_Privacy (and references cited therein). *See also, e.g.*: In Froyo, dalvik\libcore\security\src\main\java\java\security\Key.java In Gingerbread, libcore\luni\src\main\java\java\security\Key.java: /\** * {@code Key} is the common interface for all keys. * * @see PublicKey * @see PrivateKey */ public interface Key extends Serializable { … *See also, e.g.*, Android APIs for "java.security.cert," available at http://developer.android.com/reference/java/security/cert/package-summary.html. *See also, e.g.*: • Android Framework Topics for "Security and Permissions," available at http://developer.android.com/guide/topics/security/security.html

The '447 Patent	Infringed By
	• Android Framework Topics for "Security and Permissions" under "The AndroidManifest.xml File," http://developer.android.com/guide/topics/manifest/permission-element.html • Android Framework Topics for "Security and Permissions" under "The AndroidManifest.xml File," http://developer.android.com/guide/topics/manifest/application-element.html • Android Framework Topics for "The AndroidManifest.xml File," available at http://developer.android.com/guide/topics/manifest/manifest-intro.html

The '447 Patent	Infringed By
5. The method of claim 2, wherein said code identifier indicates a source of code used to define each class of said one or more classes and indicates a key associated with each class of said one or more classes.	*See* Claims 2 and 4, *supra*.

The '447 Patent	Infringed By

6. The method of claim 2, wherein the step of associating said one or more protection domains and said one or more classes based on said code identifier further includes associating said one or more protection domains and said one or more classes based on data persistently stored, wherein said data associates code identifiers with a set of one or more permissions.	*See* Claim 2, *supra*.
	
	See also, *e.g.*:
	
	In Froyo, dalvik\libcore\security\src\main\java\java\security\CodeSource.java
	In Gingerbread, libcore\luni\src\main\java\java\security\CodeSource.java:
	`/**`
	` * {@code CodeSource} encapsulates the location from where code is loaded and`
	` * the certificates that were used to verify that code. This information is used`
	` * by {@code SecureClassLoader} to define protection domains for loaded classes.`
	` *`
	` * @see SecureClassLoader`

```
 * @see ProtectionDomain
 */
public class CodeSource implements Serializable {
```

In Froyo, dalvik\libcore\security\src\main\java\java\security\Permission.java
In Gingerbread, libcore\luni\src\main\java\java\security\Permission.java:

```
/**
 * {@code Permissions} represents a {@code PermissionCollection} where the
 * contained permissions can be of different types. The permissions are
 * organized in their appropriate {@code PermissionCollection} obtained by
 * {@link Permission#newPermissionCollection()}. For permissions which do not
 * provide a dedicated {@code PermissionCollection}, a default permission
 * collection, based on a hash table, will be used.
 */
public final class Permissions extends PermissionCollection implements
        Serializable {
```

See also, *e.g.*:
In Froyo, dalvik\libcore\security\src\main\java\java\security\Key.java
In Gingerbread, libcore\luni\src\main\java\java\security\Key.java:

```
/**
 * {@code Key} is the common interface for all keys.
 *
 * @see PublicKey
 * @see PrivateKey
 */
public interface Key extends Serializable {
…
```

E.g., “Serializable” is generally understood as:

In computer science, in the context of data storage and transmission, serialization is

	the process of converting a data structure or object into a sequence of bits so that it can be stored in a file or memory buffer, or transmitted across a network connection link to be "resurrected" later in the same or another computer environment.[1] When the resulting series of bits is reread according to the serialization format, it can be used to create a semantically identical clone of the original object. For many complex objects, such as those that make extensive use of references, this process is not straightforward. http://en.wikipedia.org/wiki/Serialization (footnote omitted). Android APIs for "java.io.Serializable," available at http://developer.android.com/reference/java/io/Serializable.html: Class Overview An empty marker interface for classes that want to support serialization and deserialization based on the ObjectOutputStream and ObjectInputStream classes. Implementing this interface is enough to make most classes serializable. If a class needs more fine-grained control over the serialization process (for example to implement compatibility with older versions of the class), it can achieve this by providing the following two methods (signatures must match exactly): private void writeObject(java.io.ObjectOutputStream out) throws IOException private void readObject(java.io.ObjectInputStream in) throws IOException, ClassNotFoundException *See also, e.g.*: • Android Framework Topics for "Security and Permissions," available at http://developer.android.com/guide/topics/security/security.html • Android Framework Topics for "Security and Permissions" under "The AndroidManifest.xml File," http://developer.android.com/guide/topics/manifest/permission-element.html • Android Framework Topics for "Security and Permissions" under "The AndroidManifest.xml File," http://developer.android.com/guide/topics/manifest/application-element.html

	• Android Framework Topics for "The AndroidManifest.xml File," available at http://developer.android.com/guide/topics/manifest/manifest-intro.html

The '447 Patent	Infringed By
7. A method for providing security, the method comprising the steps of:	*See* Claim 1-pre, *supra*.
establishing one or more protection domains, wherein a protection domain is associated with zero or more permissions;	*See* Claim 1-a, *supra*.
establishing an association between said one or more protection domains and one or more sources of code; and	*See* Claim 1-a and 1-b, *supra*.
in response to executing code making a request to perform an action, determining whether said request is permitted based on a source of said code making said request and said association between said one or more protection domains and said one or more sources of code.	*See* Claim 1-c, *supra*.

The '447 Patent	Infringed By
8. The method of claim 7, wherein the step of establishing an association between said one or more protection domains and said one or more sources of code further includes establishing an association between said one or more protection	*See* Claims 2, 4, and 7, *supra*.

The ’447 Patent	Infringed By
domains and said one or more sources of code and one or more keys associated with said one or more sources of code.	

The ’447 Patent	Infringed By
9. The method of claim 8, wherein the step of establishing an association between said one or more protection domains and said one or more sources of code and said one or more keys associated with said one or more sources of code further includes establishing said association between said one or more protection domains and said one or more sources of code and said one or more keys associated with said one or more sources of code based on data persistently stored, wherein said data associates particular sources of code and particular keys with a set of one or more permissions.	*See* Claims 6 and 8, *supra*.

The ’447 Patent	Infringed By
10. A computer-readable medium carrying one or more sequences of one or more instructions, the one or more sequences of the one or more instructions including instructions which, when executed by one or	The Accused Instrumentalities include devices that store, distribute, or run Android or the Android SDK, including websites, servers, and mobile devices. These encompass a computer readable medium carrying one or more sequences of one or more instructions, the one or more sequences of the one or more instructions including instructions which, when executed by one or more processors, causes the one or more processors to perform the steps described in the claim. *See* Claim 1-pre, *supra*.

The ’447 Patent	Infringed By
more processors, causes the one or more processors to perform the steps of:	
establishing one or more protection domains, wherein a protection domain is associated with zero or more permissions;	*See* Claim 1-a, *supra*.
establishing an association between said one or more protection domains and one or more classes of one or more objects; and	*See* Claim 1-b, *supra*.
determining whether an action requested by a particular object is permitted based on said association between said one or more protection domains and said one or more classes.	*See* Claim 1-c, *supra*.

The ’447 Patent	Infringed By
11. The computer readable medium of claim 10, wherein:	*See* Claim 10, *supra*.
at least one protection domain of said one or more protection domains is associated with a code identifier;	*See* Claims 1-a and 2, *supra*.
at least one class of said one or more classes is associated with said code identifier; and	*See* Claims 1-b and 2, *supra*.
the step of establishing an association between said one or more protection domains and said one or more classes of one or more objects further includes the step of	*See* Claim 1-c and 2, *supra*.

The ’447 Patent	Infringed By
associating said one or more protection domains and said one or more classes based on said code identifier.	

The ’447 Patent	Infringed By
12. The computer readable medium of claim 11, wherein said code identifier indicates a source of code used to define each class of said one or more classes.	*See* Claim 11, *supra*.

The ’447 Patent	Infringed By
13. The computer readable medium of claim 11, wherein said code identifier indicates a key associated with each class of said one or more classes.	*See* Claims 2, 4, and 11, *supra*.

The ’447 Patent	Infringed By
14. The computer readable medium of claim 11, wherein said code identifier indicates a source of code used to define each class of said one or more classes and indicates a key associated with each class of said one or more classes.	*See* Claims 2, 4, and 11, *supra*.

The ’447 Patent	Infringed By
15. The computer readable medium of claim 14, wherein the step of	*See* Claims 6 and 14, *supra*.

The '447 Patent	Infringed By
associating said one or more protection domains and said one or more classes based on said code identifier further includes associating said one or more protection domains and said one or more classes based on data persistently stored, wherein said data associates code identifiers with a set of one or more permissions.	

The '447 Patent	Infringed By
16. A computer-readable medium carrying one or more sequences of one or more instructions, wherein the execution of the one or more sequences of the one or more instructions causes the one or more processors to perform the steps of:	The Accused Instrumentalities include devices that store, distribute, or run Android or the Android SDK, including websites, servers, and mobile devices. These encompass a computer readable medium carrying one or more sequences of one or more instructions, the one or more sequences of the one or more instructions including instructions which, when executed by one or more processors, causes the one or more processors to perform the steps described in the claim. *See* Claim 1-pre, *supra*.
establishing one or more protection domains, wherein a protection domain is associated with zero or more permissions;	*See* Claim 1 and 1-a, *supra*.
establishing an association between said one or more protection domains and one or more sources of code; and	*See* Claim 1, 1-a, and 1-b, *supra*.
in response to executing code making a request to perform an action, determining whether said request is permitted based on a source of said code making said	*See* Claim 1 and 1-c, *supra*.

The ’447 Patent	Infringed By
request and said association between said one or more protection domains and said one or more sources of code.	

The ’447 Patent	Infringed By
17. The computer readable medium of claim 16, wherein the step of establishing an association between said one or more protection domains and said one or more sources of code further includes establishing an association between said one or more protection domains and said one or more sources of code and one or more keys associated with said one or more sources of code.	*See* Claim 16, *supra*.

The ’447 Patent	Infringed By
18. The computer readable medium of claim 17, wherein the step of establishing an association between said one or more protection domains and said one or more sources of code and said one or more keys associated with said one or more sources of code further includes establishing said association between said one or more protection domains and said one or more sources of code and said one or	*See* Claim 17, *supra*.

The ’447 Patent	Infringed By
more keys associated with said one or more sources of code based on data persistently stored, wherein said data associates particular sources of code and particular keys with a set of one or more permissions.	

The ’447 Patent	Infringed By
19. A computer system comprising:	The Accused Instrumentalities include devices that run Android or the Android SDK. Devices running Android or the Android SDK are computer systems. *See* Claim 1, *supra*.
a processor;	Devices running Android and computers running the Android SDK have processors.
a memory coupled to said processor;	Devices running Android and computers running the Android SDK have a memory coupled to said processor.
one or more protection domains stored as objects in said memory, wherein each protection domain is associated with zero or more permissions;	*See* Claim 1 and 1-a, *supra*.
a domain mapping object stored in said memory, said domain mapping object establishing an association between said one or more protection domains and one or more classes of one or more objects; and	*See* Claim 1, 1-a, and 1-b, *supra*.
said processor being configured to determine whether an action requested by a particular object is permitted based on said association between said one or more protection domains and said one or more classes.	*See* Claim 1 and 1-c, *supra*.

The '447 Patent	Infringed By
20. The computer system of claim 19, wherein:	*See* Claim 19, *supra*.
at least one protection domain of said one or more protection domains is associated with a code identifier;	*See* Claim 2, *supra*.
at least one class of said one or more classes is associated with said code identifier; and	*See* Claim 2, *supra*.
said computer system further comprises said processor configured to establish an association between said one or more protection domains and said one or more classes of one or more objects by associating said one or more protection domains and said one or more classes based on said code identifier.	*See* Claim 2, *supra*.

The '447 Patent	Infringed By
21. The computer system of claim 20, wherein said code identifier indicates a source of code used to define each class of said one or more classes.	*See* Claims 2 and 20, *supra*.

The '447 Patent	Infringed By
22. The computer system of claim 20, wherein said code identifier indicates a key associated with each class of said one or more classes.	*See* Claims 2, 4, and 20, *supra*.

The ’447 Patent	Infringed By
23. The computer system of claim 20, wherein said code identifier indicates a source of code used to define each class of said one or more classes and indicates a key associated with each class of said one or more classes.	*See* Claims 2, 4, and 20, *supra*.

The ’447 Patent	Infringed By
24. The computer system of claim 20, further comprising said processor configured to associate said one or more protection domains and said one or more classes based on said code identifier by associating said one or more protection domains and said one or more classes based on data persistently stored in said computer system, wherein said data associates code identifiers with a set of one or more permissions.	*See* Claims 2, 6, and 20, *supra*.

Exhibit E

EXHIBIT G
Supplemental Infringement Contentions for US 7,426,720 (’720 Patent)

NOTE: The infringement evidence cited below is exemplary and not exhaustive. The cited examples are taken from Android 2.3 and current versions of Google’s Android websites. Oracle’s infringement contentions apply to all versions of Android having similar or nearly identical code or documentation, including past and expected future releases. Although Oracle’s investigation is ongoing, the ’720 patent is infringed by all versions of Android from Oct. 21, 2008 to the present, including Android 1.1, 1.5 (“Cupcake”), 1.6 (“Donut”), 2.0/2.1 (“Éclair”), 2.2 (“Froyo”), and 2.3 (“Gingerbread”).

The cited source code examples are taken from http://android.git.kernel.org/. The citations are shortened and mirror the file paths shown in http://android.git.kernel.org/. For example, “dalvik\vm\native\InternalNative.c” maps to “[platform/dalvik.git] / vm / native / InternalNative.c” (accessible at http://android.git.kernel.org/?p=platform/dalvik.git;a=blob;f=vm/native/InternalNative.c).

It appears that the Android git source code repository (accessible through http://android.git.kernel.org/) was created on or around Oct. 21, 2008. As such, the list of infringing Android versions may be expanded based on what Oracle learns about earlier Android versions.

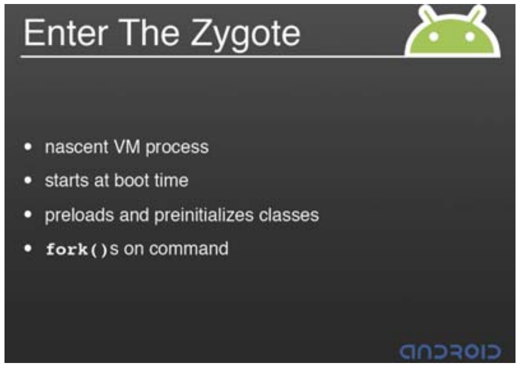
Oracle has determined that Android devices execute much of the code cited below every time the devices start up. Other cited code is invoked when a developer runs the Android Compatibility Test Suite (CTS), which Google requires manufacturers to execute to certify devices as Android-compatible.[1] The mobile device emulator that Google includes with the Android SDK[2] supports Oracle’s conclusion. The emulator displays log messages to inform developers of what is running on the virtual device. If the developer includes a logging command in part of a program, the emulator will output a log entry every time that part of the program is executed. A developer might use this feature, for example, to test whether an application starts to execute a particular section of code before failing. By adding logging commands to key portions of the Android source code cited below, building an Android system image, and loading it into Google’s emulator, Oracle determined that many of these code portions are executed even before a user can interact with a device. Thus, Android-compatible devices, when used as Google intends, execute infringing code.

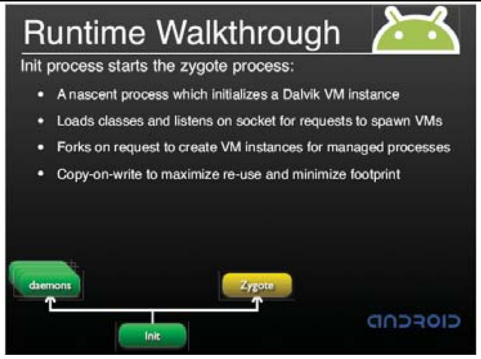
[1] http://source.android.com/compatibility/android-2.2-cdd.pdf at 10 (“To be considered compatible with Android 2.2, device implementations . . . MUST pass the most recent version of the Android Compatibility Test Suite (CTS) available at the time of the device implementation's software is completed.”).
[2] See http://developer.android.com/guide/developing/devices/emulator.html (“The Android SDK includes a virtual mobile device emulator that runs on your computer. The emulator lets you prototype, develop, and test Android applications without using a physical device. The Android emulator mimics all of the hardware and software features of a typical mobile device, except that it cannot place actual phone calls.”).

The asserted claims include system, method, and computer-readable storage medium claims. Anyone who makes, uses, offers to sell, sells, or imports a device running Android within or into the United States directly infringes the system claims. This includes Google and its downstream licensees, including device manufacturers, carriers, application developers, and end users. Similarly, anyone who engages in the above conduct with respect to storage devices containing Android code directly infringes the computer-readable storage medium claim. This includes Google and its downstream licensees, including device manufacturers, carriers, application developers, and end users. Anyone who uses a device running Android code directly infringes the method claims. This includes Google and its downstream licensees, including device manufacturers, carriers, application developers, and end users. Google induces and contributes to infringement of all asserted claims by distributing Android code with the intention that it will be executed on mobile devices. The Android code cited below necessarily infringes when it runs because its zygote process performs copy-on-write process cloning. Moreover, much of the code cited below is executed not only as applications run, but every time a device running Android starts up. Thus Android is not a staple article suitable for substantial non-infringing use. Google supplies its Android code in and from the United States.

When infringement evidence first presented with respect to one claim is referred to with respect to another, the evidence is applicable because it is not limited to a particular form of infringement.

The ’720 Patent	Infringed By
1.pre. A system for dynamic preloading of classes through memory space cloning of a master runtime system process, comprising:	The Accused Instrumentalities include systems that run Android or the Android SDK. They encompass a system running Android for dynamic preloading of classes through memory space cloning of a master runtime system process. An example of a master runtime system process is a zygote process, which creates a Dalvik virtual machine instance and which forks upon request to create new Dalvik virtual machine instances for various applications.
1.a. A processor;	A processor of a computer or smartphone running Android.
1.b. A memory	A memory of a computer or smartphone running Android.
1.c. a class preloader to obtain a representation of at least one class from a source definition provided as object-oriented program code;	Android includes a class preloader to obtain a representation of at least one class from a source definition provided as object-oriented program code. *See* Presentation slides corresponding to the Dalvik Video: “Dalvik Virtual Machine Internals, Google I/O 2008,” by Dan Bornstein, http://sites.google.com/site/io/dalvik-vm-internals/2008-05-29-Presentation-Of-Dalvik-VM-Internals.pdf (“Dalvik Presentation”), at slide 25; and corresponding Video: “Google I/O 2008 - Dalvik Virtual Machine Internals,” by Dan Bornstein, http://developer.android.com/videos/index.html#v=ptjedOZEXPM (“Dalvik

The ’720 Patent	Infringed By
	Video”), at time 13:50-15:20.  Enter The Zygote • nascent VM process • starts at boot time • preloads and preinitializes classes • `fork()`s on command (Dalvik Presentation, Slide 25) Corresponding Dalvik Video at 13:48: “What we do with the zygote, as its name implies, it’s, it comes into existence fairly early on during the boot of an Android system and its job is to load up those classes that we believe will be used across many applications. So it goes and creates, it goes and creates a heap, it goes and creates that dirty memory for all, to represent those classes and methods ….” *See also* Presentation slides corresponding to the Android Video: “Anatomy and Physiology of an Android, Google I/O 2008,” by Patrick Brady, http://sites.google.com/site/io/anatomy--physiology-of-an-android/Android-Anatomy-GoogleIO.pdf (“Android Presentation”), at slide 82; and corresponding Video: “Google I/O 2008 – Anatomy and Physiology of an Android,” by Patrick Brady, http://developer.android.com/videos/index.html#v=G-36noTCaiA (“Android Video”), at time 43:15-49:00.

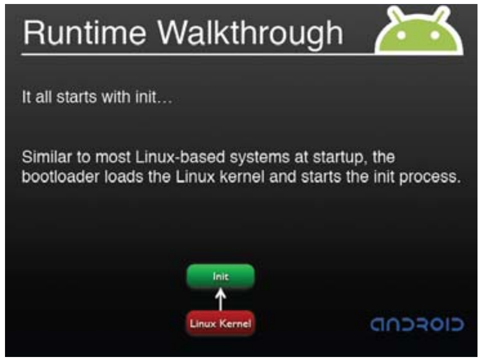
The ’720 Patent	Infringed By
	 **Runtime Walkthrough** Init process starts the zygote process: • A nascent process which initializes a Dalvik VM instance • Loads classes and listens on socket for requests to spawn VMs • Forks on request to create VM instances for managed processes • Copy-on-write to maximize re-use and minimize footprint daemons Zygote Init android (Android Presentation, Slide 82) Corresponding Android Video at 44:30: “The init process starts up a really neat process called zygote. As its name implies, zygote is really just the beginning of all of the rest of the Android platform. And so zygote is a nascent VM process that initializes a Dalvik VM and preloads a lot of its libraries….” Example source code files in base\preloaded-classes, base\core\java\com\android\internal\os\ZygoteInit.java *See, e.g.*, base\preloaded-classes. *# Classes which are preloaded by com.android.internal.os.ZygoteInit.* *# Automatically generated by frameworks/base/tools/preload/WritePreloadedClassFile.java.* *# MIN_LOAD_TIME_MICROS=1250* *# MIN_PROCESSES=10* *android.R$styleable* *android.accounts.Account* … *dalvik.system.Zygote* *java.beans.PropertyChangeEvent*

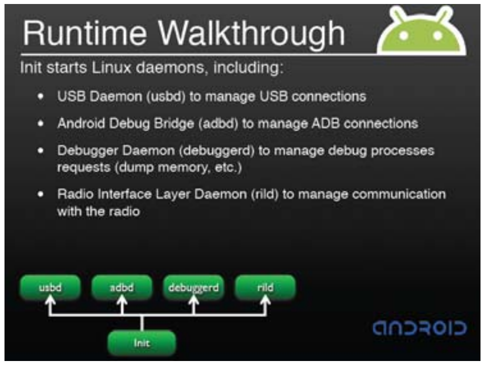
The ’720 Patent	Infringed By

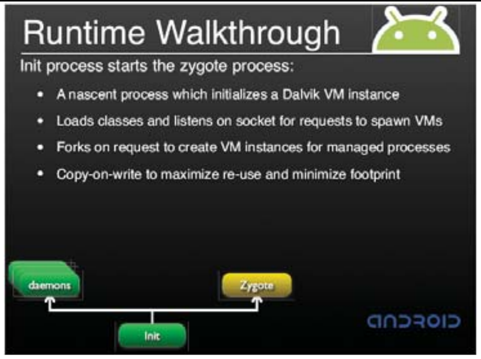
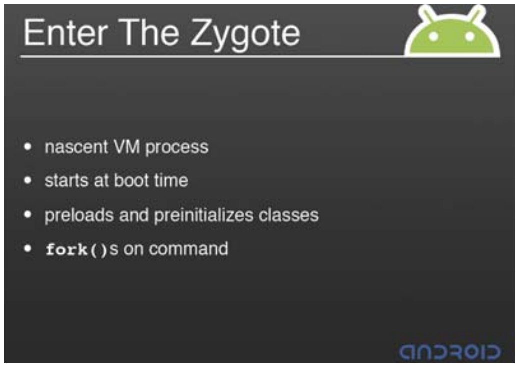
	java.beans.PropertyChangeListener
	…
	
	See, *e.g.*, base\core\java\com\android\internal\os\ZygoteInit.java.
	
	`/**`
	`   * Performs Zygote process initialization. Loads and initializes`
	`   * commonly used classes.`
	`   *`
	`   * Most classes only cause a few hundred bytes to be allocated, but`
	`   * a few will allocate a dozen Kbytes (in one case, 500+K).`
	`   */`
	`  private static void preloadClasses() {`
	`    final VMRuntime runtime = VMRuntime.getRuntime();`
	
	`    InputStream is = ZygoteInit.class.getClassLoader().getResourceAsStream(`
	`        PRELOADED_CLASSES);`
	`    if (is == null) {`
	`      Log.e(TAG, “Couldn’t find “ + PRELOADED_CLASSES + “.”);`
	`    } else {`
	`      Log.i(TAG, “Preloading classes...”);`
	`     …`
	
	`      try {`
	`        BufferedReader br`
	`          = new BufferedReader(new InputStreamReader(is), 256);`
	
	`        int count = 0;`
	`        String line;`
	`        String missingClasses = null;`
	`        while ((line = br.readLine()) != null) {`
	`          // Skip comments and blank lines.`
	`          line = line.trim();`
	`          if (line.startsWith(“#”) || line.equals(““)) {`
	`            continue;`
	`          }`

The ’720 Patent	Infringed By
	(code continued below)

```
                try {
                    if (Config.LOGV) {
                        Log.v(TAG, "Preloading " + line + "...");
                    }
                    Class.forName(line);
                    if (Debug.getGlobalAllocSize() > PRELOAD_GC_THRESHOLD) {
                        if (Config.LOGV) {
                            Log.v(TAG,
                                " GC at " + Debug.getGlobalAllocSize());
                        }
                        runtime.gcSoftReferences();
                        runtime.runFinalizationSync();
                        Debug.resetGlobalAllocSize();
                    }
                    count++;
                …
            }
        }
    }
```

The ’720 Patent	Infringed By
1.d. a master runtime system process to interpret and to instantiate the representation as a class definition in a memory space of the master runtime system process;	Android includes a master runtime system process to interpret and to instantiate the representation as a class definition in a memory space of the master runtime system process. *See*  Runtime Walkthrough — It all starts with init… Similar to most Linux-based systems at startup, the bootloader loads the Linux kernel and starts the init process. (Android Presentation, Slide 80)

The ’720 Patent	Infringed By
	Corresponding Android Video at 43:28: “Like any Linux-based or Unix-based system, at startup, the bootloader is gonna boot Linux and it’s gonna kick off the init process. This is similar to how any Linux system really starts up.”  Runtime Walkthrough Init starts Linux daemons, including: • USB Daemon (usbd) to manage USB connections • Android Debug Bridge (adbd) to manage ADB connections • Debugger Daemon (debuggerd) to manage debug processes requests (dump memory, etc.) • Radio Interface Layer Daemon (rild) to manage communication with the radio usbd adbd debuggerd rild Init android (Android Presentation, Slide 81) Corresponding Android Video at 43:41: “The first thing init is going to do on Android is start some low level, ah, processes called Linux daemons. And these are typically used to handle things like low level hardware interfaces, um, and they would sit on top of the abstraction layer and run and listen on sockets for things like USB connections or, you know, Android Debug Bridge or ADB connections, the Debugger connections and also the Radio Interface Layer daemon, which will sit on top of, um, on top of the radio baseband and interface with the baseband modem.”

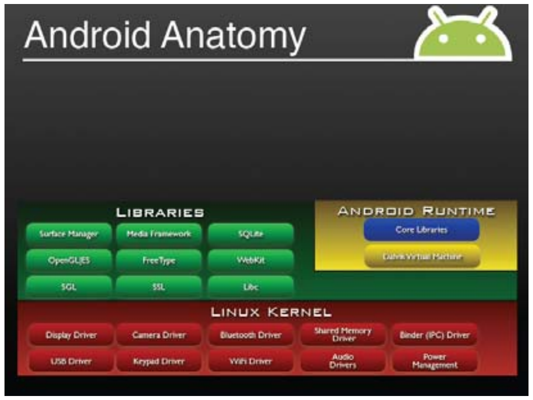
The ’720 Patent	Infringed By
	 Runtime Walkthrough Init process starts the zygote process: • A nascent process which initializes a Dalvik VM instance • Loads classes and listens on socket for requests to spawn VMs • Forks on request to create VM instances for managed processes • Copy-on-write to maximize re-use and minimize footprint daemons Zygote Init android (Android Presentation, Slide 82) Corresponding Android Video at 44:25: “Ah, after starting up the Linux daemons, and we’ll collapse those in the corner of the screen here to save some space, the init process starts up a really neat process called zygote. And as its name implies, zygote is really just the beginning of all of the rest of the Android platform. And so zygote is a nascent, ah, VM process that initializes a Dalvik VM and preloads a lot of these libraries….” *See also*  Enter The Zygote • nascent VM process • starts at boot time • preloads and preinitializes classes • `fork()`s on command (Dalvik Presentation, Slide 25)

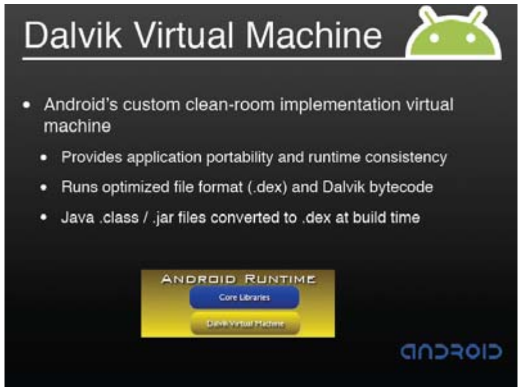
The ’720 Patent	Infringed By
	Corresponding Dalvik Video at 13:48: “What we do with the zygote, as its name implies, it’s, it comes into existence fairly early on during the boot of an Android system and its job is to load up those classes that we believe will be used across many applications. So it goes and creates, it goes and creates a heap, it goes and creates that dirty memory for all, to represent those classes and methods ….” Example source code files in base\core\jni\AndroidRuntime.cpp, base\cmds\app_process\app_main.cpp, base\core\java\com\android\internal\os\ZygotInit.java. Example code call chain, Class AppRuntime in app_main.cpp passes ZygoteInit class name to AndroidRuntime::startVm, AndroidRuntime::start(className) calls startVm, AndroidRuntime::startVm calls JNI_CreateJavaVM(), AndroidRuntime::start calls CallStaticVoidMethod(ZygoteInit className.main). *See*, *e.g.*, base\core\java\com\android\internal\os\ZygoteInit.java. */*** ** Startup class for the zygote process.* *** ** Pre-initializes some classes, and then waits for commands on a UNIX domain* ** socket. Based on these commands, forks of child processes that inherit* ** the initial state of the VM.* *** ** Please see {@link ZygoteConnection.Arguments} for documentation on the* ** client protocol.* *** ** @hide*

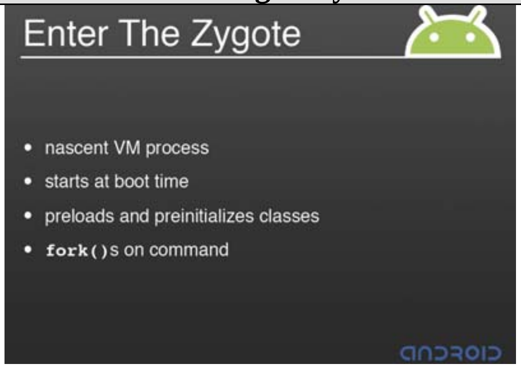
The ’720 Patent	Infringed By
	```
	*/
	…
	  public static void main(String argv[]) {
	      try {
	          VMRuntime.getRuntime().setMinimumHeapSize(5 * 1024 * 1024);

	          // Start profiling the zygote initialization.
	          SamplingProfilerIntegration.start();

	          registerZygoteSocket();
	          EventLog.writeEvent(LOG_BOOT_PROGRESS_PRELOAD_START,
	              SystemClock.uptimeMillis());
	          preloadClasses();
	          //cacheRegisterMaps();
	          preloadResources();
	          EventLog.writeEvent(LOG_BOOT_PROGRESS_PRELOAD_END,
	              SystemClock.uptimeMillis());

	          // Finish profiling the zygote initialization.
	          SamplingProfilerIntegration.writeZygoteSnapshot();

	          // Do an initial gc to clean up after startup
	          gc();

	          // If requested, start system server directly from Zygote
	          if (argv.length != 2) {
	              throw new RuntimeException(argv[0] + USAGE_STRING);
	          }

	          if (argv[1].equals("true")) {
	              startSystemServer();
	          } else if (!argv[1].equals("false")) {
	              throw new RuntimeException(argv[0] + USAGE_STRING);
	          }

	          Log.i(TAG, "Accepting command socket connections");

	          if (ZYGOTE_FORK_MODE) {
	              runForkMode();
	```

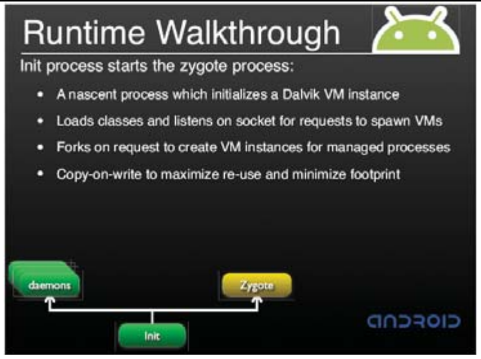
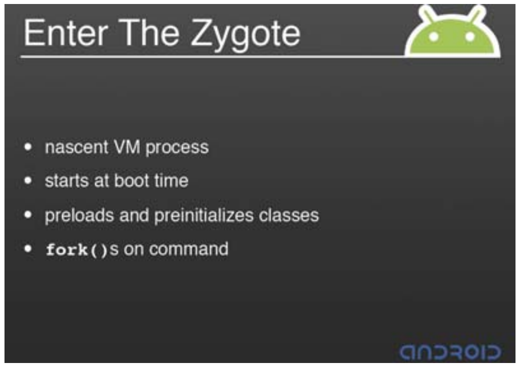
The ’720 Patent	Infringed By

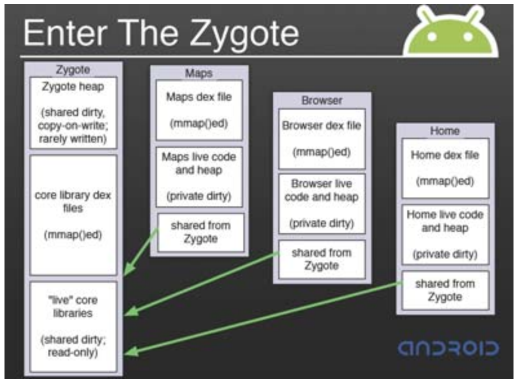
	} else {
	runSelectLoopMode();
	}
	
	closeServerSocket();
	} catch (MethodAndArgsCaller caller) {
	caller.run();
	} catch (RuntimeException ex) {
	Log.e(TAG, "Zygote died with exception", ex);
	closeServerSocket();
	throw ex;
	}
	}

1.e. a runtime environment to clone the memory space as a child runtime system process responsive to a process request and to execute the child runtime system process; and	Android includes a runtime environment to clone the memory space as a child runtime system process responsive to a process request and to execute the child runtime system process. *See*  Android Anatomy Libraries: Surface Manager, Media Framework, SQLite, OpenGL|ES, FreeType, WebKit, SGL, SSL, Libc Android Runtime: Core Libraries, Dalvik Virtual Machine Linux Kernel: Display Driver, Camera Driver, Bluetooth Driver, Shared Memory Driver, Binder (IPC) Driver, USB Driver, Keypad Driver, WiFi Driver, Audio Drivers, Power Management (Android Presentation, Slide 55) Corresponding Android Video at 33:40: “So we’ve covered the native libraries, we’ve covered everything down to the Linux kernel, and the real magic of the Android platform happens in the layers above this. And that’s what we’ll go into now, starting with the Android runtime. The Android runtime sits on top of the

The ’720 Patent	Infringed By
	libraries and Linux kernel and it provides (1) the Dalvik virtual machine and the core libraries, here written in blue, because they are exposed through the Java programming languages.”  Dalvik Virtual Machine • Android’s custom clean-room implementation virtual machine • Provides application portability and runtime consistency • Runs optimized file format (.dex) and Dalvik bytecode • Java .class / .jar files converted to .dex at build time ANDROID RUNTIME Core Libraries Dalvik Virtual Machine android (Android Presentation, Slide 56) Corresponding Android Video at 34:04: “So Dalvik virtual machine. Remember Android is not Linux. We don’t have a native windowing system. All of the applications and services that you run, will be running inside a virtual environment powered by the Dalvik virtual machine….” *See also*

The ’720 Patent	Infringed By
	 Enter The Zygote • nascent VM process • starts at boot time • preloads and preinitializes classes • `fork()`s on command (Dalvik Presentation, Slide 25) Corresponding Dalvik Video at 13:48: “What we do with the zygote, as its name implies,…when it gets a command to start up a new application, it does a normal Unix fork and then that child process becomes that target application. And the result of that is this.” *See also* claim 1.f. below.
1.f. a copy-on-write process cloning mechanism to instantiate the child runtime system process by copying references to the memory space of the master runtime system process into a separate memory space for the child runtime system process, and to defer copying of the memory space of the master runtime system process until the child runtime system process needs to modify the referenced memory space of the master runtime system process.	Android includes a copy-on-write process cloning mechanism to instantiate the child runtime system process by copying references to the memory space of the master runtime system process into a separate memory space for the child runtime system process, and to defer copying of the memory space of the master runtime system process until the child runtime system process needs to modify the referenced memory space of the master runtime system process. *See*

The ’720 Patent	Infringed By
	 Runtime Walkthrough Init process starts the zygote process: • A nascent process which initializes a Dalvik VM instance • Loads classes and listens on socket for requests to spawn VMs • Forks on request to create VM instances for managed processes • Copy-on-write to maximize re-use and minimize footprint daemons Zygote Init android (Android Presentation, Slide 82) Corresponding Android Video at 44:30: “The init process starts up a really neat process called zygote….It uses copy-on-write to maximize re-use and minimize footprint so that data structures are shared and it won’t do a full copy unless some of those data structures are to be modified.” *See also*  Enter The Zygote • nascent VM process • starts at boot time • preloads and preinitializes classes • `fork()`s on command (Dalvik Presentation, Slide 25) Corresponding Dalvik Video at 13:48:

The ’720 Patent	Infringed By
	“What we do with the zygote, as its name implies,…when it gets a command to start up a new application, it does a normal Unix fork and then that child process becomes that target application. And the result of that is this.”  Enter The Zygote Zygote Zygote heap (shared dirty, copy-on-write; rarely written) core library dex files (mmap()ed) "live" core libraries (shared dirty; read-only) Maps Maps dex file (mmap()ed) Maps live code and heap (private dirty) shared from Zygote Browser Browser dex file (mmap()ed) Browser live code and heap (private dirty) shared from Zygote Home Home dex file (mmap()ed) Home live code and heap (private dirty) shared from Zygote android (Dalvik Presentation, Slide 26) Corresponding Dalvik Video at 14:40: “So the zygote, again, has made, has made this heap of objects, it’s made this live dex structure and then each application that then starts up, instead of having its own memory for those things, it just shares it with the zygote and also with any other app that’s also on the system.” *See also* http://developer.android.com/guide/basics/what-is-android.html. “Android Runtime …The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management. Linux Kernel Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. The kernel also acts as

The ’720 Patent	Infringed By
	an abstraction layer between the hardware and the rest of the software stack.” *See also*, Lowe, Robert, Linux Kernel Process Management, April 15, 2005. Sample Chapter is provided courtesy of Sams, http://www.informit.com/articles/article.aspx?p=370047&seqNum=2&rll=1. “Copy-on-Write …In Linux, fork() is implemented through the use of copy-on-write pages. Copy-on-write (or COW) is a technique to delay or altogether prevent copying of the data. Rather than duplicate the process address space, the parent and the child can share a single copy. The data, however, is marked in such a way that if it is written to, a duplicate is made and each process receives a unique copy.” Example source code files in libcore\dalvik\src\main\java\dalvik\system\Zygote.java, dalvik\vm\native\dalvik_system_Zygote.c, linux-2.6\kernel\fork.c. Example code call chain forkAndSpecialize calls forkAndSpecializeCommon, forkAndSpecializeCommon calls fork, Linux fork process do_fork calls copy_process. *See, e.g.*, libcore\dalvik\src\main\java\dalvik\system\Zygote.java. */*** ** Forks a new Zygote instance, but does not leave the zygote mode.* ** The current VM must have been started with the -Xzygote flag. The* ** new child is expected to eventually call forkAndSpecialize()* *** ** @return 0 if this is the child, pid of the child*

The '720 Patent	Infringed By

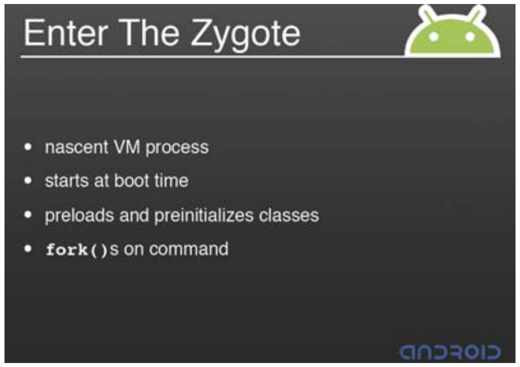
	` * if this is the parent, or -1 on error`
	` */`
	`native public static int fork();`
	
	`/**`
	` * Forks a new VM instance.  The current VM must have been started`
	` * with the -Xzygote flag. <b>NOTE: new instance keeps all`
	` * root capabilities. The new process is expected to call capset()</b>.`
	` *`
	` * @param uid the UNIX uid that the new process should setuid() to after`
	` * fork()ing and and before spawning any threads.`
	` * @param gid the UNIX gid that the new process should setgid() to after`
	` * fork()ing and and before spawning any threads.`
	` * @param gids null-ok; a list of UNIX gids that the new process should`
	` * setgroups() to after fork and before spawning any threads.`
	` * @param debugFlags bit flags that enable debugging features.`
	` * @param rlimits null-ok an array of rlimit tuples, with the second`
	` * dimension having a length of 3 and representing`
	` * (resource, rlim_cur, rlim_max). These are set via the posix`
	` * setrlimit(2) call.`
	` *`
	` * @return 0 if this is the child, pid of the child`
	` * if this is the parent, or -1 on error.`
	` */`
	`native public static int forkAndSpecialize(int uid, int gid, int[] gids,`
	`        int debugFlags, int[][] rlimits);`
	
	See, e.g., dalvik\vm\native\dalvik_system_Zygote.c.
	
	`/* native public static int forkAndSpecialize(int uid, int gid,`
	` *     int[] gids, int debugFlags);`
	` */`
	` static void Dalvik_dalvik_system_Zygote_forkAndSpecialize(const u4* args,`
	`JValue* pResult)`
	`{`
	`     pid_t pid;`
	`    pid = forkAndSpecializeCommon(args);`
	`    RETURN_INT(pid);`

The ’720 Patent	Infringed By
	```
	}
	…
	/*
	 * Utility routine to fork zygote and specialize the child process.
	 */
	static pid_t forkAndSpecializeCommon(const u4* args, bool isSystemServer)
	{
	    pid_t pid;

	    uid_t uid = (uid_t) args[0];
	    gid_t gid = (gid_t) args[1];
	    ArrayObject* gids = (ArrayObject *)args[2];
	    u4 debugFlags = args[3];
	    ArrayObject *rlimits = (ArrayObject *)args[4];
	    int64_t permittedCapabilities, effectiveCapabilities;

	    if (isSystemServer) {
	        /*
	         * Don't use GET_ARG_LONG here for now.  gcc is generating code
	         * that uses register d8 as a temporary, and that's coming out
	         * scrambled in the child process.  b/3138621
	         */
	        //permittedCapabilities = GET_ARG_LONG(args, 5);
	        //effectiveCapabilities = GET_ARG_LONG(args, 7);
	        permittedCapabilities = args[5] | (int64_t) args[6] << 32;
	        effectiveCapabilities = args[7] | (int64_t) args[8] << 32;
	    } else {
	        permittedCapabilities = effectiveCapabilities = 0;
	    }

	    if (!gDvm.zygote) {
	        dvmThrowException("Ljava/lang/IllegalStateException;",
	            "VM instance not started with -Xzygote");

	        return -1;
	    }

	    if (!dvmGcPreZygoteFork()) {
	        LOGE("pre-fork heap failed\n");
	```

The ’720 Patent	Infringed By
	dvmAbort(); *}* *setSignalHandler();* *dvmDumpLoaderStats("zygote");* *pid = fork();* *if (pid == 0) {* *int err;* */* The child process */* …. *} else if (pid > 0) {* */* the parent process */* *}* *return pid;* *}* *See, e.g.*, linux-2.6\kernel\fork.c. */** ** Ok, this is the main fork-routine.* *** ** It copies the process, and if successful kick-starts* ** it and waits for it to finish using the VM if required.* **/* *long do_fork(unsigned long clone_flags,* *unsigned long stack_start,* *struct pt_regs *regs,* *unsigned long stack_size,* *int __user *parent_tidptr,* *int __user *child_tidptr)* *{* *struct task_struct *p;* *int trace = 0;* *long nr;* … *p = copy_process(clone_flags, stack_start, regs, stack_size,*

The ’720 Patent	Infringed By

	wake_up_new_task(p, clone_flags);
	…
	tracehook_report_clone_complete(trace, regs,
	clone_flags, nr, p);
	…
	return nr;
	}

2. A system according to claim 1, further comprising: a cache checker to determine whether the instantiated class definition is available in a local cache associated with the master runtime system process.	Android includes a cache checker to determine whether the instantiated class definition is available in a local cache associated with the master runtime system process. *See*  Enter The Zygote • nascent VM process • starts at boot time • preloads and preinitializes classes • `fork()`s on command (Dalvik Presentation, Slide 25) Corresponding Dalvik Video at 13:48: “What we do with the zygote, as its name implies, it’s, it comes into existence fairly early on during the boot of an Android system and its job is to load up those classes that we believe will be used across many applications. So it goes and creates, it goes and creates a heap, it goes and creates that dirty memory for all, to represent those classes and methods….” Example source code files in dalvik\vm\oo\Class.c, dalvik\vm\native\java_lang_Class.c,

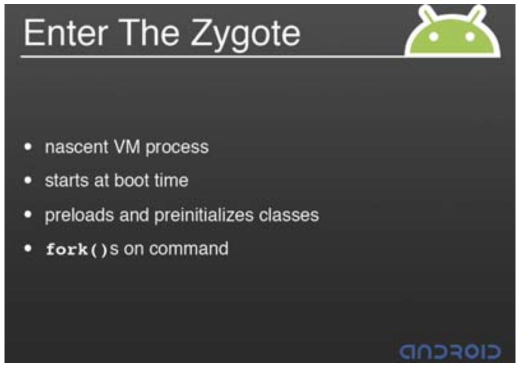
The ’720 Patent	Infringed By
	dalvik\vm\native\java_lang_VMClassLoader.c, dalvik\vm\native\dalvik_system_DexFile.c, dalvik\vm\native\InternalNative.c. Example code call chain for application classloader, Class.forName calls Class.classForName, Class.classForName calls dvmFindClassByName, dvmFindClassByName calls dvmFindClass, dvmFindClass calls dvmFindClassNoInit, dvmFindClassNoInit calls findClassFromLoaderNoInit, findClassFromLoaderNoInit calls dvmLookupClass, dvmLookupClass returns class from gDvm.loadedClasses (a table of loaded classes). Example code call chain for boot classloader, Class.forName calls Class.classForName, Class.classForName calls dvmFindClassByName, dvmFindClassByName calls dvmFindClass, dvmFindClass calls dvmFindClassNoInit, dvmFindClassNoInit calls dvmFindSystemClassNoInit, dvmFindSystemClassNoInit calls findClassNoInit, findClassNoInit calls dvmLookupClass, dvmLookupClass returns class from gDvm.loadedClasses (a table of loaded classes). *See*, *e.g.*, dalvik\vm\oo\Class.c. */* * Find the named class (by descriptor), using the specified * initiating ClassLoader. * * The class will be loaded and initialized if it has not already been. * If necessary, the superclass will be loaded. * * If the class can’t be found, returns NULL with an appropriate exception*

The ’720 Patent	Infringed By
	(continued below)

```
 * raised.
 */
ClassObject* dvmFindClass(const char* descriptor, Object* loader)
{
    ClassObject* clazz;
    clazz = dvmFindClassNoInit(descriptor, loader);
    if (clazz != NULL && clazz->status < CLASS_INITIALIZED) {
        /* initialize class */
        if (!dvmInitClass(clazz)) {
            /* init failed; leave it in the list, marked as bad */
            assert(dvmCheckException(dvmThreadSelf()));
            assert(clazz->status == CLASS_ERROR);
            return NULL;
        }
    }
    return clazz;
}


/*
 * Find the named class (by descriptor), using the specified
 * initiating ClassLoader.
 *
 * The class will be loaded if it has not already been, as will its
 * superclass.  It will not be initialized.
 *
 * If the class can’t be found, returns NULL with an appropriate exception
 * raised.
 */
ClassObject* dvmFindClassNoInit(const char* descriptor,
        Object* loader)
{
    assert(descriptor != NULL);
    //assert(loader != NULL);
    LOGVV(“FindClassNoInit ‘%s’ %p\n”, descriptor, loader);
    if (*descriptor == ‘[‘) {
        /*
         * Array class.  Find in table, generate if not found.
         */
```

The ’720 Patent	Infringed By

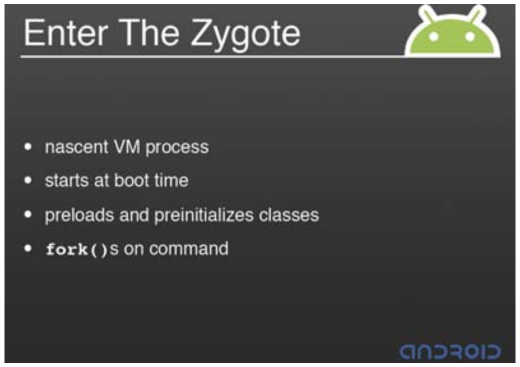
	return dvmFindArrayClass(descriptor, loader);
	} else {
	*/**
	** Regular class. Find in table, load if not found.*
	**/*
	if (loader != NULL) {
	return findClassFromLoaderNoInit(descriptor, loader);
	} else {
	return dvmFindSystemClassNoInit(descriptor);
	}
	}
	}

3. A system according to claim 2, further comprising: a class locator to locate the source definition if the instantiated class definition is unavailable in the local cache.	Android includes a class locator to locate the source definition if the instantiated class definition is unavailable in the local cache. *See* Enter The Zygote • nascent VM process • starts at boot time • preloads and preinitializes classes • `fork()`s on command (Dalvik Presentation, Slide 25) Corresponding Dalvik Video at 13:48: “What we do with the zygote, as its name implies, it’s, it comes into existence fairly early on during the boot of an Android system and its job is to load up those classes that we believe will be used across many applications. So it goes and creates, it goes and creates a heap, it goes and creates that dirty memory for all, to represent those classes and methods….”

The ’720 Patent	Infringed By

	Example source code files in
	dalvik\vm\oo\Class.c,
	dalvik\vm\native\java_lang_Class.c,
	dalvik\vm\native\java_lang_VMClassLoader.c,
	dalvik\vm\native\dalvik_system_DexFile.c,
	dalvik\vm\native\InternalNative.c.
	
	Example code call chain for application classloader or boot classloader, dvmLookupClass returns NULL, calls ClassLoader.loadClass.
	
	See, *e.g.*, dalvik\vm\oo\Class.c.
	
	`/*`
	` * Find the named class (by descriptor), using the specified`
	` * initiating ClassLoader.`
	` *`
	` * The class will be loaded and initialized if it has not already been.`
	` * If necessary, the superclass will be loaded.`
	` *`
	` * If the class can’t be found, returns NULL with an appropriate exception`
	` * raised.`
	` */`
	`ClassObject* dvmFindClass(const char* descriptor, Object* loader)`
	`{`
	`    ClassObject* clazz;`
	`    clazz = dvmFindClassNoInit(descriptor, loader);`
	`    if (clazz != NULL && clazz->status < CLASS_INITIALIZED) {`
	`        /* initialize class */`
	`        if (!dvmInitClass(clazz)) {`
	`            /* init failed; leave it in the list, marked as bad */`
	`            assert(dvmCheckException(dvmThreadSelf()));`
	`            assert(clazz->status == CLASS_ERROR);`
	`            return NULL;`
	`        }`

The ’720 Patent	Infringed By
	```
	    }
	    return clazz;
	}

	/*
	 * Find the named class (by descriptor), using the specified
	 * initiating ClassLoader.
	 *
	 * The class will be loaded if it has not already been, as will its
	 * superclass.  It will not be initialized.
	 *
	 * If the class can’t be found, returns NULL with an appropriate exception
	 * raised.
	 */
	ClassObject* dvmFindClassNoInit(const char* descriptor,
	        Object* loader)
	{
	    assert(descriptor != NULL);
	    //assert(loader != NULL);
	    LOGVV(“FindClassNoInit ‘%s’ %p\n”, descriptor, loader);
	    if (*descriptor == ‘[‘) {
	        /*
	         * Array class.  Find in table, generate if not found.
	         */
	        return dvmFindArrayClass(descriptor, loader);
	    } else {
	        /*
	         * Regular class.  Find in table, load if not found.
	         */
	        if (loader != NULL) {
	            return findClassFromLoaderNoInit(descriptor, loader);
	        } else {
	            return dvmFindSystemClassNoInit(descriptor);
	        }
	    }
	}
	```
4. A system according to claim 1, further comprising: a class resolver	Android includes a class resolver to resolve the class definition.

The ’720 Patent	Infringed By
to resolve the class definition.	*See*  Enter The Zygote • nascent VM process • starts at boot time • preloads and preinitializes classes • `fork()`s on command (Dalvik Presentation, Slide 25) Corresponding Dalvik Video at 13:48: “What we do with the zygote, as its name implies, it’s, it comes into existence fairly early on during the boot of an Android system and its job is to load up those classes that we believe will be used across many applications. So it goes and creates, it goes and creates a heap, it goes and creates that dirty memory for all, to represent those classes and methods….” Example source code files in dalvik\vm\oo\Class.c, dalvik\vm\oo\Resolve.c, dalvik\vm\native\java_lang_Class.c, dalvik\vm\native\java_lang_VMClassLoader.c, dalvik\vm\native\dalvik_system_DexFile.c, dalvik\vm\native\InternalNative.c. Example code call chain for application classloader or boot classloader, dvmLookupClass calls dvmLinkClass, dvmLinkClass calls dvmResolveClass, dvmResolveClass returns resolved class.

The ’720 Patent	Infringed By

	See, e.g., dalvik\vm\oo\Class.c.
	```
	/*
	 * Link (prepare and resolve).  Verification is deferred until later.
	 *
	 * This converts symbolic references into pointers.  It's independent of
	 * the source file format.
	 *
	 * If clazz->status is CLASS_IDX, then clazz->super and interfaces[] are
	 * holding class reference indices rather than pointers.  The class
	 * references will be resolved during link.  (This is done when
	 * loading from DEX to avoid having to create additional storage to
	 * pass the indices around.)
	 *
	 * Returns "false" with an exception pending on failure.
	 */
	bool dvmLinkClass(ClassObject* clazz)
	{
	    u4 superclassIdx = 0;
	    u4 *interfaceIdxArray = NULL;
	    bool okay = false;
	    int i;

	    assert(clazz != NULL);
	    assert(clazz->descriptor != NULL);
	    assert(clazz->status == CLASS_IDX || clazz->status == CLASS_LOADED);
	    if (gDvm.verboseClass)
	        LOGV("CLASS: linking '%s'...\n", clazz->descriptor);

	    assert(gDvm.classJavaLangClass != NULL);
	    assert(clazz->obj.clazz == gDvm.classJavaLangClass);
	    if (clazz->classLoader == NULL &&
	        (strcmp(clazz->descriptor, "Ljava/lang/Class;") == 0))
	    {
	        if (gDvm.classJavaLangClass->ifieldCount > CLASS_FIELD_SLOTS) {
	            LOGE("java.lang.Class has %d instance fields (expected at most %d)",
	```

The ’720 Patent	Infringed By
	```
	            gDvm.classJavaLangClass->ifieldCount, CLASS_FIELD_SLOTS);
	        dvmAbort();
	    }
	    if (gDvm.classJavaLangClass->sfieldCount != CLASS_SFIELD_SLOTS) {
	        LOGE("java.lang.Class has %d static fields (expected %d)",
	            gDvm.classJavaLangClass->sfieldCount, CLASS_SFIELD_SLOTS);
	        dvmAbort();
	    }
	}
	/* "Resolve" the class.
	 *
	 * At this point, clazz's reference fields may contain Dex file
	 * indices instead of direct object references.  Proxy objects are
	 * an exception, and may be the only exception.  We need to
	 * translate those indices into real references, and let the GC
	 * look inside this ClassObject.
	 */
	if (clazz->status == CLASS_IDX) {
	    if (clazz->interfaceCount > 0) {
	        /* Copy u4 DEX idx values out of the ClassObject* array
	         * where we stashed them.
	         */
	        assert(sizeof(*interfaceIdxArray) == sizeof(*clazz->interfaces));
	        size_t len = clazz->interfaceCount * sizeof(*interfaceIdxArray);
	        interfaceIdxArray = malloc(len);
	        if (interfaceIdxArray == NULL) {
	            LOGW("Unable to allocate memory to link %s", clazz->descriptor);
	            goto bail;
	        }
	        memcpy(interfaceIdxArray, clazz->interfaces, len);

	        dvmLinearReadWrite(clazz->classLoader, clazz->interfaces);
	        memset(clazz->interfaces, 0, len);
	        dvmLinearReadOnly(clazz->classLoader, clazz->interfaces);
	    }

	    assert(sizeof(superclassIdx) == sizeof(clazz->super));
	    superclassIdx = (u4) clazz->super;
	    clazz->super = NULL;
	```

The ’720 Patent	Infringed By

	`/* After this line, clazz will be fair game for the GC. The`
	` * superclass and interfaces are all NULL.`
	` */`
	`clazz->status = CLASS_LOADED;`
	
	`if (superclassIdx != kDexNoIndex) {`
	`    ClassObject* super = dvmResolveClass(clazz, superclassIdx, false);`
	`    if (super == NULL) {`
	`        assert(dvmCheckException(dvmThreadSelf()));`
	`        if (gDvm.optimizing) {`
	`            /* happens with "external" libs */`
	`            LOGV("Unable to resolve superclass of %s (%d)\n",`
	`                 clazz->descriptor, superclassIdx);`
	`        } else {`
	`            LOGW("Unable to resolve superclass of %s (%d)\n",`
	`                 clazz->descriptor, superclassIdx);`
	`        }`
	`        goto bail;`
	`    }`
	`    dvmSetFieldObject((Object *)clazz,`
	`                      offsetof(ClassObject, super),`
	`                      (Object *)super);`
	`}`
	`…`
	`/*`
	` * There are now Class references visible to the GC in super and`
	` * interfaces.`
	` */`
	`…`
	
	`/*`
	` * Done!`
	` */`
	`if (IS_CLASS_FLAG_SET(clazz, CLASS_ISPREVERIFIED))`
	`    clazz->status = CLASS_VERIFIED;`
	`else`
	`    clazz->status = CLASS_RESOLVED;`
	`okay = true;`
	`if (gDvm.verboseClass)`

The ’720 Patent	Infringed By
	```
	        LOGV("CLASS: linked '%s'\n", clazz->descriptor);

	    /*
	     * We send CLASS_PREPARE events to the debugger from here.  The
	     * definition of "preparation" is creating the static fields for a
	     * class and initializing them to the standard default values, but not
	     * executing any code (that comes later, during "initialization").
	     *
	     * We did the static prep in loadSFieldFromDex() while loading the class.
	     *
	     * The class has been prepared and resolved but possibly not yet verified
	     * at this point.
	     */
	    if (gDvm.debuggerActive) {
	        dvmDbgPostClassPrepare(clazz);
	    }

	bail:
	    if (!okay) {
	        clazz->status = CLASS_ERROR;
	        if (!dvmCheckException(dvmThreadSelf())) {
	            dvmThrowException("Ljava/lang/VirtualMachineError;", NULL);
	        }
	    }
	    if (interfaceIdxArray != NULL) {
	        free(interfaceIdxArray);
	    }
	    return okay;
	}
	```
5. A system according to claim 1, further comprising: at least one of a local and remote file system to maintain the source definition as a class file.	Android includes at least one of a local and remote file system to maintain a source definition as a class file. *See*